

INSIDE THIS ISSUE: BOB BOONSTRA INDUCTED INTO HALL OF FAME!

MacTechTM MAGAZINETM

FOR MACINTOSH PROGRAMMERS & DEVELOPERS

FEBRUARY 1995 • VOLUME 11, No. 2

In This Issue!

GETTING STARTED

Adding Your Own Class to Sprocket

APPLETALK

Yenta and the Appletalk Class Library

PROGRAMMER CHALLENGE

Symbolize

THINK TOP 10

IMPROVING THE FRAMEWORK

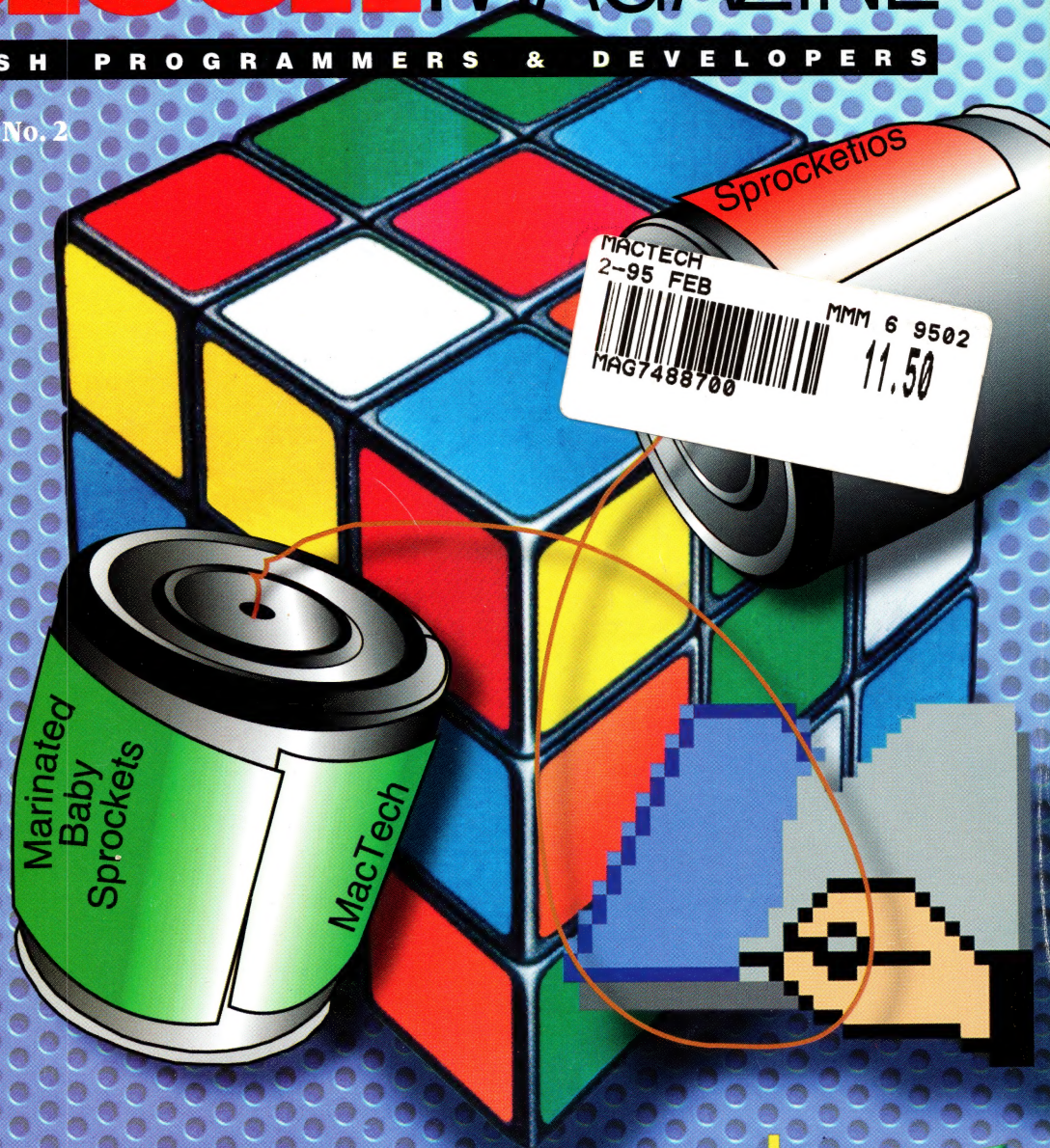
Using Low Priority Events in MacApp

UNIVERSAL RESOURCE LOCATORS

THE INQUISITIVE PROGRAMMER

A Quick Trip Into the Depths

AND MORE!



System 7.5



\$5.85 US
\$6.95 Canada
ISSN 1067-8360
Printed in U.S.A.

PROGRAPH CPX. TAKE A MAGIC CD-ROM RIDE.

Build Applications As Fast As You Can Visualize Them

Yes, *visualize* them. No ifs, ands
or C++.

There's real magic in Prograph CPX's
pictorial programming environment.
Not just another pretty interface, its
visual magic includes the actual
down and dirty of programming
itself. What's the payoff? Maximum
productivity.

Prograph CPX is the award-
winning OOP tool of choice for
client/server solutions, sophisticated
multimedia authoring and all your
development needs. And
you won't have to charm
any snakes to deliver
killer applications in less
time and with less money.

Test drive Prograph
absolutely free and
experience why it's the
#1 rapid application
development tool.

Just call 800.927.4847.

Ask for your magic CD-ROM ride.



Rapid application development.
Maximum flexibility.
And the ride of your life.

FREE
CD offer —
call for details



prograph
international

P.O. Box 2206, San Francisco, CA 94126-2206
Internet: info@prograph.com AOL: PrographUS

™ © 1995 Prograph International. All other trademarks are
the property of their respective holders.

Time Is On Your Side

Now VIP-C creates applications that are accelerated for Power Macintosh.

Now, create 68K and Power Macintosh applications faster than ever before. VIP-C's visual design, its application building aids, and its ability to let you modify your program code and interface with immediate feedback, give you a real programming edge. Read further to see why, with VIP-C, time really is on your side...

Complete Development System

Now develop with just one comprehensive system. VIP-C integrates all the tools you need to create stand-alone 68K and Power Macintosh applications—right out of the box! Develop applications in a standard language using this intelligent, full-featured Rapid Application Development (RAD) environment. No more moving from tool to tool. No more dependence on non-standard languages.

For All Levels of Programmers

VIP-C offers multiple levels of support to best fit your abilities. If you just want to type C code into its editor, you can—and VIP-C will automatically check the syntax and create a flowchart of your code. For higher-level support, VIP-C features prewritten intelligent prototypes of all the Mac Toolbox calls, high-level VIP-C Functions that simplify low-level Toolbox calls, Resource Editors that let you visually design and

create your windows, menus, dialogs, buttons, etc., and even a VIP-C Dispatcher to provide an efficient and reliable application framework—all in one complete development environment.

User Interface Generation Tools

The VIP-C Dispatcher simplifies program development by automating the main event loop. It acts as a central controller to manage program events by distributing tasks to different routines. User interface items—menus, dialogs, buttons, etc.—are created with integrated resource editors that automatically link user action to your code.

VIP-C 1.5
Everything you need to
create complete Macintosh and
Power Macintosh applications
in one box.

Powerful Prewritten Functions

Functions for the most frequently used features of the Mac interface have been prewritten and stored in a palette for instant use. With the click of a mouse you can lighten your programming workload and speed design and coding time by several orders of magnitude.

On-Line Macintosh Toolbox Calls

VIP-C accesses the complete Mac Toolbox. Search and display prototypes for any function, structure, or macro. No more rooting through multiple volumes of Inside Macintosh!

Attention Database Developers!

VIP-C users now have the power to create commercial quality, multi-user, relational databases using the new VIP Database Manager (sold separately). VIP Database Manager has the built-in functionality of the CXBase Pro database engine (recently chosen by Apple OSA for creating the eWorld content publishing software) giving you an inexpensive way to create professional databases, royalty-free!

Compiler Hotlink

If you have a favorite compiler, use it! Compile and run your application directly from a menu item in the VIP-C editor. VIP-C supports: THINK C, MPW-C, and CodeWarrior (each sold separately).



Buy VIP-C now for a Limited-Time Special Price of only \$295. Act now and get VIP Database Manager for only \$149.

VIP-C's suggested retail price is \$495. But from now until March 1, 1995, you can order VIP-C for \$295. Call Mainstay to order today.

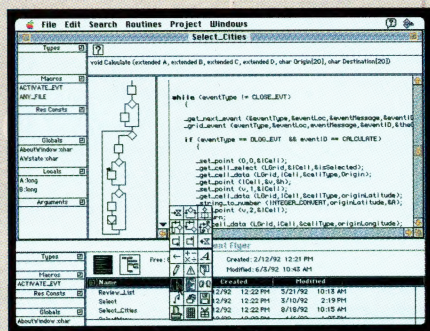
Mainstay

(805) 484-9400

(800) 484-9817 Ext. 8472

591-A Constitution Ave.
Camarillo, CA 93012
fax: (805) 484-9428

71 rue des Airébates
B-1040 Brussels, Belgium
32-2/733.97.91



VIP-C is a trademark of Mainstay. Apple, Macintosh, Power Macintosh, and MPW are registered trademarks of Apple Computer, Inc. THINK is a trademark of Symantec Corporation. CodeWarrior is a trademark of Metrowerks.

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**? If you have any questions, feel free to call us at 310/575-4343 or fax us at 310/575-0925.

DEPARTMENTS	Internet	CompuServe	AppleLink	America Online	GEne
Orders, Circulation, & Customer Service	custservice@xplain.com	71333,1063	MT.CUSTSVC	MT CUSTSVC	MACTECHMAG
Editorial	editorial@xplain.com	71333,1065	MT.EDITORIAL	MT EDITORS	
Programmer's Challenge	progchallenge@xplain.com	71552,174	MT.PROGCHAL	MT PRGCHAL	
Ad Sales	adsales@xplain.com	71552,172	MT.ADSALES	MT ADSALES	
Accounting	accounting@xplain.com	—	—	—	
Marketing	marketing@xplain.com	—	—	—	
Press Releases	pressreleases@xplain.com	—	—	—	
General	info@xplain.com	71333,1064	MACTECHMAG	MacTechMag	
Online support area	ftp://ftp.netcom.com/pub/xp/xplain	<i>type</i> GO MACTECHMAG	<i>see Third Parties:</i> Third Parties (H-O)	<i>use keyword:</i> MACTECHMAG	—

MacTECH MAGAZINE

Publisher Emeritus • David Williams
Editor-in-Chief/Publisher • Neil Ticktin
Editor • Scott T Boyd
Associate Editor • Mary Elaine Califf
Editorial Assistant • John Kawakami
Advertising Executive • Ruth Subrin
Art Director • Judith Chaplin, Chaplin & Assoc.

XPLAIN CORPORATION

VP Finance & Operations • Andrea J. Sniderman
Customer Service • Al Estrada
Software Engineer • Don Bresee
Accounting Assistant • Brian Shin
Administrative Assistant • Susan Pomrantz
Board of Advisors • Blake Park, Alan Carsrud



AUTHORS & REGULAR CONTRIBUTORS

MacTech Magazine is grateful to the following individuals who contribute on a regular basis. We encourage others to share the technology. We are dedicated to the distribution of useful programming information without regard to Apple's developer status. For information on submitting articles, ask us for our **writer's kit** which includes the terms and conditions upon which we publish articles.

Richard Clark
 General Magic
 Mountain View, CA

David R. Mark
 M/MAC
 Arlington, VA

Mike Scanlin
 Programmer's Challenge
 Mountain View, CA

Chris Espinosa
 Apple Computer, Inc.
 Cupertino, CA

Jordan Mattson
 Apple Computer, Inc.
 Cupertino, CA

Symantec Technical Support Group
 THINK Division
 Eugene, OR

The names MacTech, MacTech Magazine, MacTutor and the MacTutorMan logo are registered trademarks of Xplain Corporation. All contents are copyright 1991-1994 by Xplain Corporation. All rights reserved. Trademarks appearing in MacTech Magazine remain the property of the companies that hold license.

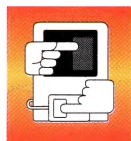


Printed on recycled paper.



MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 1617 Pontius Avenue, 2nd Floor, Los Angeles, CA 90025-9555. Voice: 310/575-4343, FAX: 310/575-0925. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Second Class postage is paid at Los Angeles, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 250055, Los Angeles, CA 90025-9555.



GETTING STARTED

Adding Your Own Class to Sprocket 7

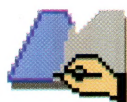
Part 1 – A Linked List Class — *By Dave Mark*



APPLETALK

Yenta and the Appletalk Class Library 16

ChatterBox for the aspiring MOOSE — *By Eric Rosé*



IMPROVING THE FRAMEWORK

Using Low Priority Events in MacApp 41

Fixing a minor bug gets your priorities straight — *By Harry Haddon*



THE INQUISITIVE PROGRAMMER

A Quick Trip Into the Depths 44

ResError Considered Harmful? — *By Steve Jasik*



PROGRAMMERS' CHALLENGE

Symbolize 48

— *By Mike Scanlin*



THINK TOP 10 63

— *By Mark B. Baldwin and Steve Howard, Symantec Technical Support*



UNIVERSAL RESOURCE LOCATORS 66

— *By Scott T Boyd and John Kawakami*



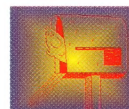
EDITOR'S PAGE

4



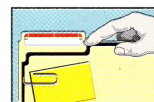
NEWSBITS

69



DIALOG BOX

68



THE CLASSIFIEDS

71



MAIL ORDER STORE

73



**ADVERTISER &
PRODUCT INDEX**

79



TIPS & TIDBITS

80



By Scott T Boyd, Editor



The Editor's Viewpoint

LEARNING FROM OTHERS' MISTAKES

If the world didn't know about Intel's Pentium processor before, they sure do now. Intel spent millions last year flying their logo around TV sets to establish their brand image. If only they'd known how much cheaper it would be to have Andrew Grove, Intel President and CEO, apologize on the Internet. Never mind that many on the net took offense that Intel would choose when and whether to replace a customer's chip. Intel not only seemed reluctant to own up to their problem (they did, after all, wait several months until someone else discovered it independently, to mention it), but much of what they said sounded more like they were upset that they got caught.

Those of us Macintosh supporters smugly rubbing our hands together with glee and thinking, "Goody, goody! Intel's in trouble," may have another thing coming. Intel hurt their reputation, right? Was it just my imagination, or was that a full-length Pentium ad shown on MacNeil/Lehrer, Night Line, and every other major news program? Will the market remember Intel's mistake? Or will they remember over a full week of coverage? Will they remember the names Intel and Pentium? and that Intel eventually took care of their customers? Even with a chargeback of tens of millions of dollars to pay for the chip replacements, Intel may have scored an advertising coup like we've never seen before.

FOR THE MORBIDLY CURIOUS

After reading Intel's white paper (<http://www.intel.com>), IBM's position paper (<http://www.ibm.com>), and Intel's rebuttal to IBM's paper, there seems to be no disagreement about the source of the bug (missing entries in a lookup table caused by a flawed script written to download entries into a hardware PLA (Programmable Lookup Array)). They also seem to agree about the worst-case impact on any single operation (inaccuracies can occur starting with the 4th significant decimal digit).

How politics affected the presentation of the data shows in the assumptions each party made. Intel assumed that the average spreadsheet user about 1000 floating point divides on any given day, and that numbers are uniformly distributed. They also used spreadsheets from all around Intel to determine the probability of occurrence for bad number combinations.

IBM, on the other hand, figured that the average spreadsheet user would spend about 15 minutes a day recalculating, and would get one divide per 16,000 instructions when recalculating. At 90MHz, that's about 4687 divides/second, or 4.2M per day. They also assert that all bit patterns are not equally probable. They created random numbers in a variety of common decimal patterns, and used them to create numerators and denominators. Their

observations led them to believe that one out of every 100 million divisions might lead to bad results.

Intel says one error every 27,000 years. IBM says every 24 days. Do you hear the axes grinding? Bottom line? Intel takes a \$35M-\$70M charge against earnings to give customers what they want (and now they *know* what's inside and that they *want* one).

HUNGRY AND SLEEPY?

I recently got a Connectix QuickCam (the \$99 all-seeing eyeball that plugs into a Mac's serial port). I plugged it in, installed some software, and all of a sudden my modem was a problem. Not that there's anything technically wrong with the modem or their software – perhaps I should back up and explain a bit.

The camera works with pretty much any QuickTime software. One particularly interesting application is CU-SeeMe, a free application from Cornell University which supports real-time multi-party videoconferencing on the Internet. Even a 14.4 connection will get you video, but it'll leave you hungry for more – much more bandwidth, that is. Four of us in three different parts of the country got online with three different cameras and a VCR. Even though we didn't do much more than watch each other smile at the camera, we had so much fun that we were all wondering how to beg, borrow, or steal more bandwidth (know any internet providers who want to trade Macintosh code for a frame relay connection?). We don't know whether it will help our virtual businesses (it sure didn't help our sleep patterns), but we have little doubt that videoconferencing will one day look no more surprising in a home office than a copier, fax machine, or a Macintosh.

QUOTABLE

"We're certainly not going to replace your Pentium chip just so you can play Doom!"
– Intel Pentium hotline staffer

"Maybe you ought to consider a Macintosh this Christmas."
– Wall Street Journal 15 Dec 1994

"I suppose it is the corrected chip that will be called RePentium."
– Peter G. Neumann

"Intel – changing the way people think about floating point."
– excerpt from a speech, originally intended as a compliment,
as reported by Jörg Brown

FOOD FOR THOUGHT

Right in the middle of a MacNeil/Lehrer News Hour montage of Pentium clips, I saw Apple's Graphing Calculator spinning a 3D parabolic equation. They didn't realize they were showing Apple's Power Macintosh, not a Pentium box. Do you think they might have been able to see the bugs had it really been running on a Pentium?



Software Developers:

It's Midnight. Do You Know Where Your Software is?

"Quark/QSS has chosen HASP and MachASP to protect QuarkXpress® in our most demanding markets, because we believe that Aladdin's products meet the high standards of reliability, compatibility and security required for these markets..."

John MacMonagle
*Production Manager,
Quark/QSS*

Bringing software into the world is a little like bringing up children. You always know where they start, but you seldom know where they'll end up. These days, with illegal use of software so common, concerned developers have good reason to worry about the products of their labor. That's where MachASP comes in.



Like a responsible babysitter, MachASP accompanies your software wherever it goes. With MachASP there, your software won't run out of control. Without MachASP, in fact, your software won't run at all.

For developers, MachASP provides the highest level of security and reliability. For legitimate users, MachASP is a friendly and transparent solution. Once connected, they won't even feel it's there.

And if your child wants to play with its friends, a single Net-MachASP lets it run free around a local area network. But always under your supervision and control.

Get serious about software protection. Since 1984, nearly one million HASP keys have enabled thousands of PC & Mac software developers, in more than 60 countries, to protect their software. To find out why MachASP is considered the best product in the market, order your MachASP Developer's Kit today.

1-800-223-4277

ALADDIN

The Professional's Choice

North America **Aladdin Software Security Inc.**
Tel: (800) 223 4277, 212-564 5678
Fax: 212-564 3377
E-mail: sales@hasp.com

Intl Office **Aladdin Knowledge Systems Ltd.**
Tel: 972-3-537 5795, Fax: 972-3-537 5796
AppleLink: ALADDIN.KNOW
E-mail: aladdin@aladdin.co.il

United Kingdom **Aladdin Knowledge Systems UK Ltd.**
Tel: 0753-622266, Fax: 0753-622262

France **Aladdin France SA**
Tel: 1 40 85 98 85, Fax: 1 41 21 90 56

© Aladdin Knowledge Systems Ltd. 1985-1994 (12.94) PowerPC is a trademark of Motorola. Macintosh is a trademark of Apple Inc.

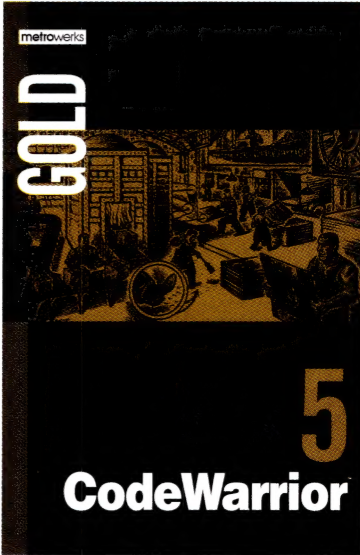
member of



■ **Australia** Conlab 3 8985685 ■ **Benelux** Aladdin Benelux 080 782098 ■ **Czech** Atlas 2 766085 ■ **Chile** Micrologica 2 222 1388
■ **Denmark** Berendsen 39 577100 ■ **Egypt** Zeineldein 2 3604632 ■ **Finland** ID-Systems 0 870 3520 ■ **Germany** CSS 201 278804
■ **Greece** Unibrain 1 6856320 ■ **Italy** Partner Data 2 26147380 ■ **Japan** Athena 3 58 213284 ■ **Korea** Dae-A 2 848 4481 ■ **Mexico** SiSoft 5 5439770
■ **New Zealand** Training 4 5666014 ■ **Poland** Systherm 61 480273 ■ **Portugal** Futurmatica 1 4116269 ■ **South Africa** D Le Roux, 11 886 4704
■ **Spain** PC Hardware, 3 4493193 ■ **Switzerland** Opag 61 7169222 ■ **Taiwan** Teco 2 555 9676 ■ **Turkey** Mikrobeta 312 467 7504

metrowerks


The fastest way to write 68K and Power Macintosh code.



Twelve Compilers!

CodeWarrior Gold includes twelve compilers and CodeWarrior Bronze includes four compilers. With our Power Mac-hosted 68K cross compilers, you can build both 68K and Power Mac apps on the Power Mac. Conversely, with our 68K Mac-hosted Power PC cross compilers you can build 68K and Power Mac apps on a 68K Mac.

Note: CodeWarrior compilers are shipped as fat binaries!



Native IDEs!

The CodeWarrior integrated development environment (IDE) is native on both the 68K Mac and Power Mac. The same easy-to-use environment is used to program in C, C++ and Pascal. Note: CodeWarrior IDEs are also fat binaries.

Get To Market Faster!

CodeWarrior compilers build your 68K and Power Mac apps at +200,000 lines/minute on a Power Mac 8100, with comparably high rates on less powerful Macs!

Two Free Updates!

CodeWarrior is updated 3 times a year on CD-ROM. Purchase either version of CodeWarrior and you are entitled to receive two future updates free of charge.



New Features!

- MPW tools included for both the 68k and Power PC
- C++ Templates supported in the compilers
- Global optimization improved for the Power PC C/C++ compilers
- Conditional breakpoints and expression evaluation added for the debugger

Two CodeWarrior versions to choose from!
Choose the CodeWarrior that suits your needs:

	BRONZE 68K Mac	GOLD Power Mac/68K Mac
Power Mac-hosted Power Mac IDE:		
Power Mac-hosted C/C++ generating PPC code		•
Power Mac-hosted Pascal generating PPC code		•
Power Mac-hosted 68K Mac IDE:		
Power Mac-hosted C/C++ generating 68K code	•	•
Power Mac-hosted Pascal generating 68K code	•	•
68K Mac-hosted Power Mac IDE:		
68K Mac-hosted C/C++ generating PPC code		•
68K Mac-hosted Pascal generating PPC code		•
68K Mac-hosted 68K Mac IDE:		
68K Mac-hosted Pascal generating 68K code	•	•
68K Mac-hosted C/C++ generating 68K code	•	•
MPW Tools:		
Power Mac-hosted C/C++ generating PPC code		•
Power Mac-hosted C/C++ generating 68K code	•	•
68K Mac-hosted C/C++ generating PPC code		•
68K Mac-hosted C/C++ generating 68K code	•	•
PowerPlant Application Framework	•	•
PowerPlant Shared Library for Power Mac		•
Source-level debugger for Power Mac		•
Source-level debugger for 68K Mac	•	•
Constructor Visual Interface Editor	•	•
Selected Apple Developer Tools	•	•
30-day money-back guarantee	•	•
3,000 pages of On-line Documentation and much, much more.	•	•



CodeWarrior 5 Gold
For Power Macintosh & 68K
Macintosh development.

LNG 0070.....\$399

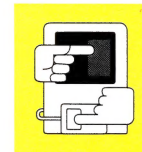
CodeWarrior 5 Bronze
For 68K Macintosh development.

LNG 0068.....\$99

Requires: Motorola 68020 or higher or PowerPC 601 processor;
8MB RAM; System 7.1; CD-ROM Drive



CALL 1-800-377-5416
International (419) 281-1802
Fax (419) 281-6883



By Dave Mark, MacTech Magazine Regular Contributing Author

Adding Your Own Class to Sprocket

Part 1 – A Linked List Class

Last month, we finally got into our new (and ever evolving) framework, Sprocket. In the next few columns, we're going to create a new set of classes designed to add functionality to Sprocket. This month, we'll create and test our new classes and next month we'll step through the process of adding the classes to Sprocket.

I know, I know. Last month I said we were going to take a closer look at Sprocket's window classes. Bear with me. Every time I dig into this C++ framework stuff, my perspective changes and I get a new sense of the direction in which I should be heading. We'll get to the window classes eventually...

A LINKED LIST CLASS

Every framework needs some sort of linked list class. You might want to maintain a list of CDs or your favorite movies. You might be building some sort of network server that maintains a list of network service requests. Whatever your need, there are probably a million ways to design a linked list class that fits the bill. In some cases, you'll adopt a general approach, designing a set of classes intended for many different applications. In other cases, you'll have a specific functional or performance need and you'll design a

class that might not be of much use to anyone else, but will solve your problem.

Dave Falkenburg (Sprocket's daddy) and I were chatting a few weeks ago about some of the features Dave envisioned for Sprocket's future. One of these features centered around a method for keeping track of your application's documents. As an example, when the user quits your application, you need to step through each of your open documents, calling each document's close method. Some applications solve this problem by stepping through the window list maintained by the system for every open application. Besides the technical mumbo-jumbo you have to go through to maintain compatibility with older versions of the MacOS, there are two basic problems with this approach. Some of your windows may not be associated with a document, and some of your documents may require more than a single window.

The linked list classes we're going to explore this month were designed specifically to maintain a list of document object pointers. As you'll see, I tried to generalize the linked list classes so that you could use them to store pointers to any objects you like, but the member functions (aka, methods) were designed with document management in mind. We'll get into the specifics of maintaining a list of document object pointers next month when we add the classes to Sprocket. This month we're going to enter the classes, then take them for a test drive.

THE LIST TESTER PROJECT

This month's source code was tested using both CodeWarrior and Symantec C++. Pick your favorite compiler and build a new iostream-based project. Figure 1 shows my CodeWarrior project window. Be sure you add the three libraries shown. If you intend on generating PowerPC code, you'll need to swap the two 68K-specific libraries for those appropriate to the PowerPC.

Figure 2 shows the Symantec C++ version of the ListTester project window. If you are using Symantec C++, be sure to add the three libraries CPlusLib, ANSI++, and IOStreams to your project. You can have this done automatically by selecting "C++ IOStreams Project" from the list of project types that appear when you select New Project from the THINK Project Manager's File menu.

ListTester.u			
File	Code	Data	
▼ Segment 1	2K	216	
Link.cp	492	48	• ▶
LinkedList.cp	1464	56	• ▶
main.cp	1082	112	• ▶
▼ Segment 2	126K	12K	
ANSI (4i) C++.68K.Lib	83046	2930	▶
ANSI (4i) C.68K.Lib	44736	10003	▶
CPlusPlus.lib	1320	32	▶
6 file(s)	129K	12K	

Figure 1. The CodeWarrior version of the ListTester project.

ListTester.m	
Name	Code
▼ Segment 2	3198
CPlusLib	1776
Link.cp	220
LinkedList.cp	838
main.cp	360
▼ Segment 3	28544
ANSI++	28540
▼ Segment 4	30500
IOStreams	30496
Totals	62820

Figure 2. The Symantec C++ version of the ListTester project.

As you can see from the two figures, you'll be adding 3 source code files to the project. In addition, you'll be creating 2 additional include files, bringing the grand total to 5. The next five sections contain the source code for each of these five files. Type in the code (assuming you haven't already downloaded it), save it under the appropriate file name, and add each of the 3 ".cp" files to the project.

MAIN.CP

```
#include <iostream.h>
#include "LinkedList.h"

void CountAndDisplayLinks( TLinkedList *listPtr );

int main()
{
    TLinkedList    *listPtr;
    char           *string;
    char           *s1 = "Frank Zappa",
                  *s2 = "Violent Femmes",
                  *s3 = "Jane Siberry";

    listPtr = new TLinkedList;

    listPtr->CreateAndAddLink( s1 );
    listPtr->CreateAndAddLink( s2 );
    listPtr->CreateAndAddLink( s3 );
```

```
CountAndDisplayLinks( listPtr );

cout << "-----\n";

string = (char *)listPtr->GetNthLinkObject( 2UL );
listPtr->FindAndDeleteLink( string );

CountAndDisplayLinks( listPtr );

return 0;
}
```

CountAndDisplayLinks

```
void CountAndDisplayLinks( TLinkedList *listPtr )
{
    unsigned long    counter, numLinks;
    char             *string;

    numLinks = listPtr->CountLinks();

    cout << "This list has ";
    cout << numLinks;
    cout << " links...\n";

    for ( counter = 1; counter <= numLinks; counter++ )
    {
        cout << "Link #" << counter << ": ";
        string = (char *)listPtr->GetNthLinkObject( counter );

        cout << string << "\n";
    }
}
```

LINKEDLIST.H

```
#ifndef _LINKEDLIST_
#define _LINKEDLIST_

#ifdef _LINK_
#include "Link.h"
#endif

const OSErr kLinkedList_LinkNotFoundErr = -2;
const OSErr kLinkedList_CouldNotDeleteLinkErr = -3;

class TLinkedList
{
public:
    TLinkedList();
    ~TLinkedList();

    virtual OSErr CreateAndAddLink(void *objectPtr);
    virtual OSErr FindAndDeleteLink(void *objectPtr);
    virtual unsigned long CountLinks();
    virtual void *GetNthLinkObject(unsigned long linkIndex);

protected:
    virtual void DeleteAllLinks();
    TLink *FindLink( void *objectPtr );
    virtual OSErr DeleteLink( TLink *linkPtr );

    TLink *fFirstLinkPtr;
    TLink *fLastLinkPtr;
};

#endif
```

LINKEDLIST.CP

```
#include "LinkedList.h"
#include "Link.h"

TLinkedList::TLinkedList()
{
    fFirstLinkPtr = nil;
    fLastLinkPtr = nil;
}
```



```

TLinkedList::~TLinkedList()
{
    DeleteAllLinks();
}

```

TLinkedList::CreateAndAddLink

```

OS_ERR TLinkedList::CreateAndAddLink( void *objectPtr )
{
    TLink    *newLinkPtr;

    newLinkPtr = new TLink( objectPtr );

    if ( newLinkPtr == nil )
        return kLink_BadLinkErr;

    if ( fFirstLinkPtr == nil )
        fFirstLinkPtr = newLinkPtr;

    if ( fLastLinkPtr != nil )
        fLastLinkPtr->SetNextLink( newLinkPtr );

    newLinkPtr->SetPrevLink( fLastLinkPtr );
    newLinkPtr->SetNextLink( nil );

    fLastLinkPtr = newLinkPtr;

    return noErr;
}

```

TLinkedList::FindAndDeleteLink

```

OS_ERR TLinkedList::FindAndDeleteLink( void *objectPtr )
{
    TLink    *foundLinkPtr;

    foundLinkPtr = FindLink( objectPtr );

    if ( foundLinkPtr == nil )
        return kLinkedList_LinkNotFoundErr;
    else
        return DeleteLink( foundLinkPtr );
}

```

TLinkedList::CountLinks

```

unsigned long TLinkedList::CountLinks()
{
    TLink    *currentLinkPtr;
    unsigned long    numLinks;

    numLinks = 0;
    currentLinkPtr = fFirstLinkPtr;

    while ( currentLinkPtr != nil )
    {
        numLinks++;
        currentLinkPtr = currentLinkPtr->GetNextLink();
    }

    return numLinks;
}

```

TLinkedList::GetNthLinkObject

```

void    *TLinkedList::GetNthLinkObject( unsigned long
linkIndex )
{
    TLink    *currentLinkPtr;
    unsigned long    numLinks, curLinkIndex;

    numLinks = CountLinks();

    if ( (linkIndex < 1) || (linkIndex > numLinks) )
        return nil;

    curLinkIndex = 0;
    currentLinkPtr = fFirstLinkPtr;

    for ( curLinkIndex=1; curLinkIndex<linkIndex; curLinkIndex++)
        currentLinkPtr = currentLinkPtr->GetNextLink();

    return currentLinkPtr->GetObjectPtr();
}

```

TLinkedList::DeleteAllLinks

```

void TLinkedList::DeleteAllLinks()
{
    TLink    *currentLinkPtr, *nextLinkPtr;

    currentLinkPtr = fFirstLinkPtr;

    while ( currentLinkPtr != nil )
    {
        nextLinkPtr = currentLinkPtr->GetNextLink();
        delete currentLinkPtr;
        currentLinkPtr = nextLinkPtr;
    }

    fFirstLinkPtr = nil;
    fLastLinkPtr = nil;
}

```

TLinkedList::FindLink

```

TLink    *TLinkedList::FindLink( void *objectPtr )
{
    TLink    *currentLinkPtr;

    currentLinkPtr = fFirstLinkPtr;

    while ( currentLinkPtr != nil )
    {
        if ( currentLinkPtr->GetObjectPtr() == objectPtr )
            return currentLinkPtr;

        currentLinkPtr = currentLinkPtr->GetNextLink();
    }

    return nil;
}

```

TLinkedList::DeleteLink

```

OS_ERR    TLinkedList::DeleteLink( TLink *linkPtr )
{
    if ( linkPtr == nil )
        return kLinkedList_CouldNotDeleteLinkErr;

    if ( linkPtr == fFirstLinkPtr )
        fFirstLinkPtr = linkPtr->GetNextLink();
    else
        linkPtr->GetPrevLink()->
            SetNextLink( linkPtr->GetNextLink() );

    if ( linkPtr == fLastLinkPtr )
        fLastLinkPtr = linkPtr->GetPrevLink();
    else
        linkPtr->GetNextLink()->
            SetPrevLink( linkPtr->GetPrevLink() );

    return noErr;
}

```

LINK.H

```

#ifndef    _LINK_
#define    _LINK_

```

```

#include <types.h>

```

```

const short kLink_BadLinkErr = -1;

```

```

class    TLink
{

```

```

public:

```

```

    TLink( void *objectPtr );
    ~TLink();
    virtual void    SetPrevLink( TLink *prevLinkPtr )
        { fPrevLinkPtr = prevLinkPtr; }
    virtual void    SetNextLink( TLink *nextLinkPtr )
        { fNextLinkPtr = nextLinkPtr; }
    virtual TLink    *GetPrevLink()
        { return fPrevLinkPtr; }
    virtual TLink    *GetNextLink()
        { return fNextLinkPtr; }
    virtual void    *GetObjectPtr()
        { return fObjectPtr; }
}

```

```

class TLink

```



```

protected:
    TLink      *fPrevLinkPtr;
    TLink      *fNextLinkPtr;
    void       *fObjectPtr;
};

#endif

```

LINK.CP

```

#include "Link.h"

TLink::TLink( void *objectPtr )
{
    fObjectPtr = objectPtr;
    fPrevLinkPtr = nil;
    fNextLinkPtr = nil;
}

TLink::~TLink()
{
}

```

RUNNING LINKTESTER

Once all your code is typed in and the appropriate files are added to your project, you're ready to go. When you run ListTester, an iostream console window will appear, showing the following output:

```

This list has 3 links...
Link #1: Frank Zappa
Link #2: Violent Femmes
Link #3: Jane Siberry
-----
This list has 2 links...
Link #1: Frank Zappa
Link #2: Jane Siberry

```

Now let's make some sense out of all this. LinkedList.h contains the declaration of a linked list class, namely TLinkedList. We'll start all our class names off with the letter T to stay compatible with Sprocket. It's just a convention and doesn't affect the code in any way. Pure semantics. LinkedList.cp contains the definitions of the TLinkedList member functions.

A TLinkedList consists of a series of TLink objects, all linked together via pointers. A TLinkedList object is an entire linked list, while a TLink is a single link in the list. Link.h contains the declaration of the TLink class, and Link.cp contains the definitions of the TLink member functions.

If this is your first time working with linked lists, take some time to read up on the basics. *Learn C on the Macintosh* will get you started, but it doesn't really get into any theory. Once you understand the basic linked list mechanism, you'll want to explore some of the more sophisticated data structures and the algorithms that make them work. There are a lot of good books out there. My personal favorite is Volume 1 ("Fundamental Algorithms") of Donald Knuth's series *The Art of Computer Programming*.

ListTester starts by creating a new TLinkedList object, then adds three new links to the list. The links contain three C

text strings, but could easily handle a document object or any other block of data. Once we add the three links to the list, we call a routine that displays the contents of the list.

Next, we call a member function to delete the second link in the list, then display the list again. That's about it. Let's take a look at the source code.

MAIN.CP

main.cp starts off by including <iostream.h>, which gives it access to cout and the rest of the iostream library. We also include LinkedList.h to give us access to the members of the TLinkedList class.

```

#include <iostream.h>
#include "LinkedList.h"

```

CountAndDisplayLinks() walks through a linked list and displays the strings embedded in the list.

```
void CountAndDisplayLinks( TLinkedList *listPtr );
```

main() starts off by creating a new TLinkedList object. Notice that the TLinkedList constructor doesn't take any parameters.

```

int main()
{
    TLinkedList      *listPtr;
    char             *string;
    char             *s1 = "Frank Zappa",
                    *s2 = "Violent Femmes",
                    *s3 = "Jane Siberry";

```

```
listPtr = new TLinkedList;
```

Next, we call the CreateAndAddLink() member function to add our three text strings to the list. We then call CountAndDisplayLinks() to walk through the list and display the contents.

```

listPtr->CreateAndAddLink( s1 );
listPtr->CreateAndAddLink( s2 );
listPtr->CreateAndAddLink( s3 );

CountAndDisplayLinks( listPtr );

cout << "-----\n";

```

Next, we'll retrieve the second object in the list, so we can delete it by calling FindAndDeleteLink(). There are a few interesting things to note here. First, notice that we had to typecast the value returned by GetNthLinkObject() to a (char *). Each TLink features a data member which points to the data associated with that link. As you'll see, the TLink stores the data as a (void *). The advantage of this strategy is that it lets you store any type of data you like in the list. You can even mix data types in a single list. The catch is, you have to know what the data type is when you retrieve it. If you plan on mixing data types, you can start each data block off with a flag that tells you its type, or you can add a data member to the TLink class (or, better yet, to a class you derive from TLink) that specifies the type of data stored in a link.

The second point of interest here is the fact that we deleted the data from the list using the data itself instead of

specifying its position in the list. In other words, we said, go find the string "Violent Femmes" and delete it, rather than, delete the 2nd item in the list. There are definitely pros and cons to this approach. Since these classes were defined to handle documents, this approach should work just fine. A more sophisticated strategy might assign a serial number to each link, then delete the link by specifying its serial number. Since document object pointers will be unique, our approach should be OK. The true test will come down the road as we add more sophisticated document handling capabilities to Sprocket.

```
string = (char *)listPtr->GetNthLinkObject( 2UL );
listPtr->FindAndDeleteLink( string );
```

Finally, we redisplay the list to verify the link's deletion.

```
CountAndDisplayLinks( listPtr );

return 0;
}
```

CountAndDisplayLinks() is pretty straightforward. We first call CountLinks() to find out how many links are in the list, then loop through that many calls to GetNthLinkObject().

```
void CountAndDisplayLinks( TLinkedList *listPtr )
{
    unsigned long    counter, numLinks;
    char             *string;

    numLinks = listPtr->CountLinks();

    cout << "This list has ";
    cout << numLinks;
    cout << " links...\n";

    for ( counter = 1; counter <= numLinks; counter++ )
    {
        cout << "Link #" << counter << ": ";
        string = (char *)listPtr->GetNthLinkObject( counter );
        cout << string << "\n";
    }
}
```

LINKEDLIST.H

LinkedList.h contains the declaration of the TLinkedList class. As we did in our last C++ column, we start the .h file off with some code that prevents us from multiply declaring the class in case a .cp file includes this file and also includes another .h file that includes this file.

```
#ifndef _LINKEDLIST_
#define _LINKEDLIST_

#ifndef _LINK_
#include "Link.h"
#endif
#endif
```

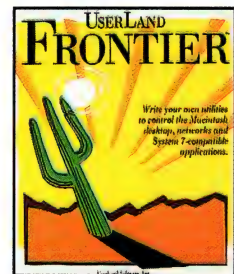
These two constants are error codes returned by various TLinkedList member function. Though our little test program didn't test for these errors, our Sprocket code definitely will. Until Sprocket supports true C++ exception handling, our error checking will consist of checking the return codes returned by member functions and bubbling the errors up to the routine that must deal with the error.

MacWeek's Score: AppleScript ♦ ♦ ♦ UserLand Frontier™ ♦ ♦ ♦ ♦!

UserLand Frontier™ isn't just a language; it's a full-featured script development and runtime environment. It gives you the tools you need to get the job done in record time!

FRONTIER 3.0 FEATURES

- ♦ **Outline-based script editor**
 - Full search and replace
 - No 32K limit!
- ♦ **Integrated Database**
 - Scripts can communicate/share data
- ♦ **Full-featured debugger**
 - Breakpoints, step in/out
 - View/modify any script values
- ♦ **Hundreds of built-in verbs**
 - Full control of file system and OS
 - No osax needed for most scripts
- ♦ **Full multi-threaded environment**
 - Full-time agents; built in script scheduler
- ♦ **Professional programming language**
 - UserTalk is easy to learn and use; based on familiar languages
 - Case statement, continue/break, optional parameters, and more...
- ♦ **Completely OSA compatible**
 - Record, edit and run scripts in any OSA language
 - Execute UserTalk scripts from any OSA environment



PowerMac Native
Version Available

Find out what thousands of users have known since 1992: Frontier is the premier scripting system for the Macintosh™!

To Order Or For More Information Call
800/845-1772

Userland Software, Inc.
555 Bryant Street
Palo Alto, CA 94301

```
const OSErr kLinkedList_LinkNotFoundErr = -2;
const OSErr kLinkedList_CouldNotDeleteLinkErr = -3;
```

The TLinkedList class features a constructor, a destructor, and four public member functions. CreateAndAddLink() creates a new TLink, embeds the objectPtr in the link, then adds the link at the end of the list. FindAndDeleteLink() walks through the list till it finds a link containing a pointer that matches objectPtr. When the match is found, the link is deleted. CountLinks() returns the number of links in the list. GetNthLinkObject() walks down the list and returns the objectPtr embedded in the Nth link in the list.

As we discussed in an earlier column, marking the destructor and other member functions as virtual allows the proper member function to be called when a new class is derived from this class and a base class pointer holds a pointer to the derived class. For more details, look up virtual destructors in your favorite C++ book.

```
class TLinkedList
{
public:
    TLinkedList();
    ~TLinkedList();

    virtual CreateAndAddLink(void *objectPtr);
    virtual OSErr FindAndDeleteLink(void *objectPtr);
    virtual OSErr
```



```
virtual unsigned long CountLinks();
virtual void *GetNthLinkObject(unsigned long linkIndex);
```

The protected members are not intended for public consumption. Instead, they are used internally by the linked list member functions.

```
protected:
    virtual void      DeleteAllLinks();
    TLink            *FindLink( void *objectPtr );
    virtual OSErr     DeleteLink( TLink *linkPtr );

    TLink            *fFirstLinkPtr;
    TLink            *fLastLinkPtr;
};

#endif
```

LINKEDLIST.CP

Since the TLinkedList member functions work with both TLinkedList and TLink members, we need to include both .h files.

```
#include "LinkedList.h"
#include "Link.h"
```

The TLinkedList constructor sets the pointers to the first and last links in the list to nil. By the way, nil is defined in <Types.h>. Also, note that all data members start with the letter f (again, just a convention).

```
TLinkedList::TLinkedList()
{
    fFirstLinkPtr = nil;
    fLastLinkPtr = nil;
}
```

The destructor deletes all the links in the list.

```
TLinkedList::~~TLinkedList()
{
    DeleteAllLinks();
}
```

CreateAndAddLink() creates a new TLink, then uses the TLink member functions SetPrevLink() and SetNextLink() to connect the link into the linked list. Each link features a prev and a next pointer, pointing to the previous and next links in the list. These two pointers make our linked list a doubly-linked list. We won't get into the advantages and disadvantages of doubly versus singly-linked lists here. Suffice it to say that we definitely could have solved our problem any number of ways.

```
OSErr TLinkedList::CreateAndAddLink( void *objectPtr )
{
    TLink *newLinkPtr;

    newLinkPtr = new TLink( objectPtr );

    if ( newLinkPtr == nil )
        return kLink_BadLinkErr;

    if ( fFirstLinkPtr == nil )
        fFirstLinkPtr = newLinkPtr;

    if ( fLastLinkPtr != nil )
        fLastLinkPtr->SetNextLink( newLinkPtr );
```

```
newLinkPtr->SetPrevLink( fLastLinkPtr );
newLinkPtr->SetNextLink( nil );

fLastLinkPtr = newLinkPtr;

return noErr;
}
```

FindAndDeleteLink() calls FindLink() to find the link in the list, then deletes the link if it was found.

```
OSErr TLinkedList::FindAndDeleteLink( void *objectPtr )
{
    TLink *foundLinkPtr;

    foundLinkPtr = FindLink( objectPtr );

    if ( foundLinkPtr == nil )
        return kLinkedList_LinkNotFoundErr;
    else
        return DeleteLink( foundLinkPtr );
}
```

CountLinks() starts off at the beginning of the list (at the link pointed to by fFirstLinkPtr), then uses GetNextLink() to walk down the list, counting links until we get to the last link, which will always have a next pointer of nil.

```
unsigned long TLinkedList::CountLinks()
{
    TLink *currentLinkPtr;
    unsigned long numLinks;

    numLinks = 0;
    currentLinkPtr = fFirstLinkPtr;

    while ( currentLinkPtr != nil )
    {
        numLinks++;
        currentLinkPtr = currentLinkPtr->GetNextLink();
    }

    return numLinks;
}
```

GetNthLinkObject() first checks to be sure the requested link is actually in the list.

```
void *TLinkedList::GetNthLinkObject( unsigned long
linkIndex )
{
    TLink *currentLinkPtr;
    unsigned long numLinks, curLinkIndex;

    numLinks = CountLinks();

    if ( (linkIndex < 1) || (linkIndex > numLinks) )
        return nil;
```

Once we know we've got a valid link, we'll step through the list the proper number of times to get to the requested link, then call GetObjectPtr() to retrieve the object pointer.

```
curLinkIndex = 0;
currentLinkPtr = fFirstLinkPtr;

for ( curLinkIndex=1; curLinkIndex<linkIndex; curLinkIndex++)
    currentLinkPtr = currentLinkPtr->GetNextLink();

return currentLinkPtr->GetObjectPtr();
}
```

DeleteAllLinks() steps through the list and deletes every link in the list. Notice that we save the next pointer before we

Eddy Award Winner for Best New Developer Tool
– MacUser Editors Choice Awards, 1993

"A distinct improvement over ResEdit."
– MacTech / MacTutor

"Resorcerer's data template system is amazing!"
– Bill Goodman, author of Compact Pro

"Nuke ResEdit! Resorcerer is mission-critical for us."
– Dave Winer, Userland Frontier

"The color pixel editors are wonderful! A work of art!"
– Dave Winzler, author of Microseeds Redux

"Every Macintosh developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensable in its own creation!"
– Doug McKenna, author of Resorcerer

"...a wealth of time-saving tools."
MacUser Review, Dec. 1992



RESORCERER[®]

Version 1.2.4

The Resource Editor for the Macintosh Wizard

ORDERING INFO

Needs: ≥Mac Plus, ≥ Sys 4.2, 1MB
Likes: ≥Mac Plus, ≥ Sys 7.0, 2MB
32-bit clean, AU/X compatible

Price: \$256 (decimal)
(Educational, quantity, or
other discounts available)

Includes: 500 page manual
60-day Money-Back Guarantee
Domestic UPS ground shipping

Payment: Check, PO's, or Visa/MC

Extras (call us):
COD, FedEx, UPS Blue/Red,
International Shipping

Downloadable Demos/Updaters:
AppleLink: Software Sampler
AOL: Software Libs/Development
CompuServe: MACDEV/Tools
or call us.

New 1.2 Features:

- New 'cicn', 'ppat', 'crsr', 'acur', 'pltt', 'clut' editors
 - Powerful icon family editing (all 9 icon types)
 - Color pixel anti-aliasing, dithering, and lots more
 - Complete 'PICT' disassembly and reassembly
 - Resource sorting; ROM resource browsing
 - 120 template field parsing types now supported
 - New insertion & deletion template field types
 - Text-only 'PICT' resources
 - Lots of improvements throughout
-
- Easier, faster, more Mac-like, and more productive than ResEdit
 - Safer memory-based, not disk-file-based, design and operation
 - All file information and common commands in one easy-to-use window
 - Compares resource files, and even **edits your data forks** as well
 - Visible, accumulating, editable scrap
 - Searches and opens/marks/selects resources by text content
 - Makes global resource ID or type changes easily and safely
 - Builds resource files from simple Rez-like scripts
 - Most editors DeRez directly to the clipboard
 - All graphic editors support screen-copying or partial screen-copying
 - Hot-linking Value Converter for editing 32 bits in a dozen formats
 - Its own 32-bit List Mgr can open and edit very large data structures
 - Templates can pre- and post-process any arbitrary data structure
 - Includes nearly 200 templates for common system resources
 - TEMPLs for Installer, MacApp, QT, Help, AppleEvent, OCE, GX, etc.
 - Full integrated support for editing color dialogs and menus
 - Try out balloons, 'ictb's, lists and popups, even create C source code
 - Integrated single-window Hex/Code Editor, with patching, searching
 - Editors for cursors, versions, pictures, bundles, and lots more
 - Well-designed, helpful developer tools being added all the time
 - Relied on by thousands of Macintosh developers around the world

MATHEMÆSTHETICS, INC.

P.O. Box 298 • Boulder • CO • 80306-0298 • USA

Phone: (303) 440-0707 • Fax: (303) 440-0504

AppleLink/AmericaOnline: RESORCERER • Internet: resorcerer@aol.com

delete the link so we don't delete the next pointer along with it.

```
void TLinkedList::DeleteAllLinks()
{
    TLink      *currentLinkPtr, *nextLinkPtr;

    currentLinkPtr = fFirstLinkPtr;

    while ( currentLinkPtr != nil )
    {
        nextLinkPtr = currentLinkPtr->GetNextLink();
        delete currentLinkPtr;
        currentLinkPtr = nextLinkPtr;
    }

    fFirstLinkPtr = nil;
    fLastLinkPtr = nil;
}
```

FindLink() steps through the list (does this stepping code look familiar?) and returns the current TLink if its object pointer matches the parameter. If the entire list is searched and no match is found, FindLink() returns nil.

```
TLink      *TLinkedList::FindLink( void *objectPtr )
{
    TLink      *currentLinkPtr;

    currentLinkPtr = fFirstLinkPtr;

    while ( currentLinkPtr != nil )
    {
        if ( currentLinkPtr->GetObjectPtr() == objectPtr )
            return currentLinkPtr;

        currentLinkPtr = currentLinkPtr->GetNextLink();
    }
    return nil;
}
```

DeleteLink() deletes the specified link, then reconnects the previous link with the link that follows the deleted link.

```
OSErr      TLinkedList::DeleteLink( TLink *linkPtr )
{
    if ( linkPtr == nil )
        return kLinkedList_CouldNotDeleteLinkErr;

    if ( linkPtr == fFirstLinkPtr )
        fFirstLinkPtr = linkPtr->GetNextLink();
    else
        linkPtr->GetPrevLink()->
            SetNextLink( linkPtr->GetNextLink() );

    if ( linkPtr == fLastLinkPtr )
        fLastLinkPtr = linkPtr->GetPrevLink();
    else
        linkPtr->GetNextLink()->
            SetPrevLink( linkPtr->GetPrevLink() );

    return noErr;
}
```

LINK.H

Link.h includes <types.h> to give it access to the definition of nil.

```
#ifndef      _LINK_
#define      _LINK_
```

```
#include <types.h>
```

The TLink class includes a single error code.

```
const short kLink_BadLinkErr = -1;
```

In addition to the constructor and destructor, the TLink class includes two setter and three getter functions. A setter function sets a data member to a specified value. A getter function returns the value of a data member. Though you can mark the data members as public, it's a better idea to limit access to them to getter and setter functions. By convention, getter and setter functions are defined in-line, rather than cluttering up the .cp file.

```
class      TLink
{
public:
    TLink( void *objectPtr );
    ~TLink();
    virtual void      SetPrevLink( TLink *prevLinkPtr )
        { fPrevLinkPtr = prevLinkPtr; }
    virtual void      SetNextLink( TLink *nextLinkPtr )
        { fNextLinkPtr = nextLinkPtr; }
    virtual TLink      *GetPrevLink()
        { return fPrevLinkPtr; }
    virtual TLink      *GetNextLink()
        { return fNextLinkPtr; }
    virtual void      *GetObjectPtr()
        { return fObjectPtr; }

protected:
    TLink      *fPrevLinkPtr;
    TLink      *fNextLinkPtr;
    void      *fObjectPtr;
};

#endif
```

LINK.CP

Since our five getters and setters were defined in the header file, the file Link.cp is pretty skimpy. The constructor initializes the link's data members and the destructor does nothing at all.

```
#include "Link.h"

TLink::TLink( void *objectPtr )
{
    fObjectPtr = objectPtr;
    fPrevLinkPtr = nil;
    fNextLinkPtr = nil;
}

TLink::~TLink()
{
}
```

TILL NEXT MONTH...

I love data structures. They are the backbone of any software program. Once you master the linked list, you can move on to binary trees (which are my personal favorites), then to hash tables and the like. I'll try to find an excuse to implement some of these structures as classes in a future column. In the meantime, experiment with these classes. Think about what you'd need to do to build a list of document objects using Sprocket. Where would you create the TLinkedList object? Would you need a global TLinkedList pointer? Where would you create the TLinks? Where would you put the code that deletes the TLinks? We'll address all of these issues next month...



There are **Billions** of reasons
to protect your software.

**You only
need
one**

reason to protect with Sentinel:

It's the worldwide standard in software protection.



Piracy is the greatest threat to the world's software industry. Developers lose billions in sales to software piracy each year. Protect your software and get all the revenue you deserve.

More developers rely on Sentinel®, from Rainbow, than any other software protection in the world.

And for good reason. Sentinel performs where it matters most: leading the industry in technology, quality, reliability and support.

So when it's time to protect your Macintosh application - and you need just-in-time delivery - protect with Sentinel. Protect with confidence.

Call Rainbow for a Sentinel Developer's Kit and a FREE copy of "The Sentinel Guide to Securing Software."



Call Today!
1-800-852-8569



RAINBOW
TECHNOLOGIES

SENTINEL
Securing the future of software

9292 Jeronimo Road • Irvine, CA 92718 • Tel: 714/454-2100 • Fax: 714/454-8557 • Applelink D3058 • International offices:
U.K. (44) 1932 570066 • France (33) 1 47 38 21 21 • Germany (49) 89 32 17 98 0 • North Carolina 800/843-0413
©1994 Rainbow Technologies, Inc. Sentinel is a registered trademark of Rainbow Technologies, Inc. All other product names are trademarks of their respective owners.



By Eric Rosé, cp3a@andrew.cmu.edu

Yenta and the Appletalk Class Library

ChatterBox for the aspiring MOOSE

A HISTORICAL NOTE...

The infancy of my Macintosh programming life was spent writing network software and hacking Appletalk code; the scars have mostly healed. IAC and the Communication Toolbox have made network software easier to write, but there is still a burden on the programmer to manage zone and node identification, as well as actually sending the data – especially if you want to provide a different interface than the PPC Browser, or if you want to communicate invisibly (eg. an e-mail driver which should seamlessly perform the task of finding other people it can send messages to).

This first part of this article describes the Appletalk Class Library – a modular and extensible approach to Appletalk communication using Think C++. The ACL is a set of objects which encapsulate tasks like initializing Appletalk, getting lists of zones, looking up and confirming addresses, and sending data between nodes. These objects are fully asynchronous, and provide a way for you to specify completion routines which can

move memory (a big plus!). The second part of this article shows how to use these classes in conjunction with a number of TCL-like but fully C++ interface classes to build “Yenta” – a “chat” application in the spirit of Ron Hafernik’s “ChatterBox” program (yay Ron!).

TO SPARE YOU THE TROUBLE...

The ACL itself consists of about 2800 lines of code, which clearly prohibits me from going into all of it in detail in this article. Much of the Appletalk code within the ACL has been seen before either in IM:IAC or DTS programs such as GetZoneList or DMZ. My general approach will be to give the declaration for each class I discuss and describe how it can be used, only showing method definitions when they illustrate some interesting or important concepts and programming strategies. The interface classes which I use to build Yenta are also of my own devising and consist of about 9000 lines of code. Since the focus of this article is on the ACL, I will not discuss them except when absolutely necessary. All the code for the ACL and the Yenta application is available from Xplain.

APPLETALK REVIEW

Anyone who wants the real lowdown on Appletalk network topology and protocols should read the second edition of “Inside Appletalk” or Michael Peirce’s book “Programming with Appletalk”. These details are not important for the development of the ACL, so this review will talk primarily about what kind of information we have to keep track of in order to communicate on an Appletalk network.

Every Appletalk network consists of zero or more zones; each zone is represented by a unique name. If there is no zone at all, the name “*” is used as a default. Once you have the zone name you can perform lookups, confirmations, etc. Every

Eric Rosé – Having recently escaped from a Masters program in Electrical and Computer Engineering, Eric Rosé writes software for medical treatment systems by day. By night Eric does Macintosh consulting and pursues his interests in software development environments, neat interface hacks, and on-line substitutes for the books which are overflowing his apartment. You can reach him for comment at cp3a@andrew.cmu.edu. His not-so-secret aspiration is to achieve the esteemed designation of MOOSE (Macintosh Object Oriented Software Engineer).

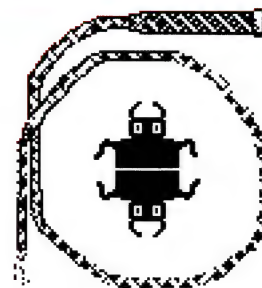
Gives you the Information to Program your Best!



Information

The Debugger V2 & MacNosy

by Steve Jasik



Control

The Debugger is a low and high-level symbolic Debugger that runs in a full multi-window Macintosh environment. You can trace program execution, view the values of variables, etc. of both 68K and PowerPC programs.

MacNosy is a global interactive disassembler that enables one to recover the source code of any Mac application, resource file or the ROM.

When you compare features of the different debuggers, note that *only one* has all the below features to help you get your job done, and *only one* has MacNosy to help you debug any program in a full system (6.0x or System 7.x) environment symbolically!

It is the *only* debugger to use the MMU to protect your CODE resources and the rest of the system from the program you are debugging. With MMU Protection you can find errors when they happen, not millions of instructions later! (Macintoshes with 68030 CPUs only).

The Debugger is the debugger of choice at: Adobe, Aldus, Claris, Electronic Arts, Kodak, Metrowerks, etc.



An example of a structured data display window

Its Features Include:

- **Symbolic Debugging** of any Macintosh program, ROM, or code resource (DRVRs, XCMDs, INITs, PDEFs, 4DEXs ..)
- **Source level debugging for Metrowerks & MPW** compiled programs (C++, C, Pascal, Fortran, ...), and an Incremental Build System with instant Link for superfast development.
- **Object Inspector for MacApp 3 programs**
- **Source level debugging of Think C™ projects**
- **Includes a program (CoverTest) to interactively do Code Coverage analysis for SQA testing, etc.**
- **Simultaneous Symbolic debugging of multiple "tasks"**
- **Fast Software Watchpoint** command to find clobbered variables
- **Sophisticated error check algorithms** such as Trap Discipline (Argument Checking), Handle Zapping, Heap Scramble and Heap Check to detect program errors before they become disasters
- **Structured display** of data (hypertext) with user definable structures while debugging
- **Conditional breakpoints** to help filter out redundant information
- **Continuous Animated Step Mode** to watch your program execute instruction by instruction
- **Detailed symbolic disassembly for both 680x0 and PowerPC** with symbol names, labels, cross ref maps, - make it possible to ferret out the secrets of the ROM, etc.
- **"Training Wheels"** for the PowerPC disassembler to help you learn the opcodes

The Debugger V2 & MacNosy: \$350

Runs on all Macs. Call For Group prices or Updates. Visa/MC Accepted.

Available from: Jasik, APDA, Frameworks or ComputerWare (800-326-0092).

Jasik Designs • 343 Trenton Way, Menlo Park, California 94025 • (415) 322-1386

Internet: macnosy@netcom.com • Applelink: D1037

'visible' entity within a zone has a unique name which is specified by an object name, a type, and the zone name; the commonly used format is name:type@zone. All network entities also have a numerical address made up of a node number (the ID of the computer they are on), a socket number (a number between 1 and 255 which specifies a socket on the computer) and a zone number. There may be multiple zone numbers assigned to each zone name. Some things you can do with network addresses are 1) formally register them on the network (making them visible); 2) deregister them (making them invisible); 3) look for addresses matching a certain name or pattern; and 4) confirm an address which you obtained earlier.

The ACL uses the PPC Toolbox (IM:IAC, ch 11) to do its data transmission. The PPC Toolbox extends the 'address' of a network entity by adding to the "name:type@zone" identifier a theoretically infinite number of ports - each described by a unique "pname:ptype" pair. For the purpose of simplicity, the ACL uses the same 'name' for both the location name and the port name, although you can still specify different location and port types if you want.

Asynchronicity is a vitally important aspect of Appletalk communication. Ideally all communication will be asynchronous so that we can be doing node lookups and confirmations while we are sending large amounts of data. To determine when such a task completes, you either have to poll a flag in the task record, or assign a completion routine which is limited in power because it can't move memory; generally they simply set global variables which are polled by the application. The ACL provides an object (CPPPeriodicTask) which acts as a wrapper for periodic tasks, and a second class (CPPPeriodicTaskManager) which performs all of the polling operations for you. All you have to do is specify a method or routine to be run when the task completes, hand it to a CPPPeriodicTaskManager object and let things take care of themselves. Because your method is not called at interrupt time, as would a traditional completion routine, you can feel free to move as much memory as you want to.

OOP(s)!

Enough talk of what has to be done; let's talk about how to do it! All of the nifty features we've discussed above are implemented in the ACL in three abstract base classes and nine concrete classes. To build on a firm foundation, we'll look at the abstract classes first.

LAYING THE FOUNDATION

CPPObject

All of the objects in the ACL have a single abstract base class called CPPObject (the declaration for which is shown below). Besides providing a common ancestor, this class gives every object the potential for making an exact copy of it (Clone) or returning its name as a c-string to anyone who asks (ClassName). We will see the utility of this feature later.

```
class CPPObject {
public:
    CPPObject (void);
    virtual ~CPPObject (void);

    virtual char *ClassName (void);
    virtual CPPObject *Clone (void);
};
```

Asynchronicity

Directly descended from CPPObject is CPPPeriodicTask - an object which encapsulates the data and methods needed to maintain an asynchronous task. Associated with CPPPeriodicTask is CPPTaskManager, which maintains a list of all periodic tasks and is responsible for making sure that each task is called at least as often as it needs to be. Here are declarations for these classes. I elaborate on them below.

```
typedef void (*CompletionProc)(CPPObject *TheTask);

class CPPPeriodicTask : public CPPObject {
public:
    Boolean isRunning;
    Boolean hasCompleted;
    Boolean deleteWhenDone;

    CPPPeriodicTask (CPPTaskManager *TaskManager,
                    long minPeriod = 120,
                    Boolean deleteWhenCompleted = TRUE);
    ~CPPPeriodicTask (void);
    virtual char *ClassName (void);

    // Setter and accessor methods
    long GetPeriod (void);
    long GetTimesCalled (void);
    void SetPeriod (long newPeriod);
    OSErr TaskError (void);
    void SetCompletionProc (CompletionProc NewProc);

    // user-specific methods
    virtual Boolean NeedsToRun (void);
    virtual void DoPeriodicAction (void);
    virtual void DoCompletedAction (void);
protected:
    OSErr callResult;
    CPPTaskManager *ourManager;
    CompletionProc completion;
private:
    long minimumPeriod;
    long lastCalled;
    long timesCalled;
};

class CPPTaskManager : public CPPObjectList {
public:
    CPPTaskManager (void);
    ~CPPTaskManager (void);
    virtual char *ClassName (void);

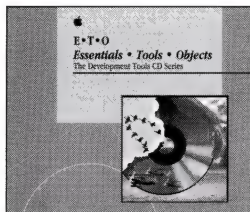
    Boolean AddPeriodicTask (CPPPeriodicTask *TheTask);
    Boolean RemovePeriodicTask (CPPPeriodicTask *TheTask,
                               Boolean disposeOfTask);
    long HowManyTasksOfType (char *ClassName);
    void RunPeriodicTasks (void);
    short HowManyPeriodicTasks (void);
};
```

CPPTaskManager (PTM) is subclassed off of an object which maintains a list of CPPObjects. Since CPPObject is the base class of the entire ACL, this list can hold any object in the ACL; in this case the list only contains CPPPeriodicTask objects, or any object descended from them.

The AddPeriodicTask and RemovePeriodicTask



Streamline your application development with powerful tools from Apple.®

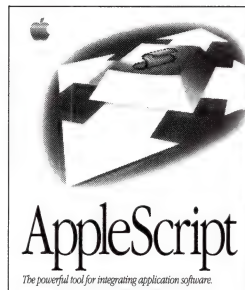


E.T.O. Essentials • Tools • Objects

E.T.O. is a deluxe collection of core programming tools for Macintosh® developers. It contains an integrated set of development environments, compilers, debuggers, application frameworks, and testing tools that you can use to streamline your application development. E.T.O. is sold by annual subscription.

E.T.O.:
Essentials • Tools • Objects
Complete New Subscriber
Package
M0895LL/C \$1095.00

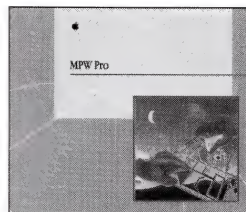
Annual E.T.O. Subscription
Renewal (3 issues)
R0076LL/B \$400.00



AppleScript™

AppleScript is a powerful and versatile tool that makes it easy to integrate the functionality of scriptable applications into a single, seamless, custom solution.

AppleScript Software
Development Toolkit v.1.1
R01752Z/C \$199.00

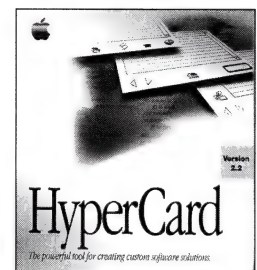


MPW® Pro

MPW Pro contains a comprehensive set of tools you can use to develop Macintosh software in C, C++, or assembly language for 68K-based and Power PC-based Macintosh systems. It also includes MacApp®, an object-oriented application framework for accelerating application development.

MPW Pro
R0505LL/C \$495.00

MPW Development System to
MPW Pro upgrade
R0582LL/A \$295



HyperCard® 2.2

HyperCard is an ideal development tool for a wide range of applications. HyperCard works like a deck of electronic index cards. You can easily integrate 24-bit color, PICTs, QuickTime™ and AppleScript to create custom software solutions and manage information.

HyperCard 2.2
M2365LL/A (U.S.) \$99.00
M2365Z/A (Int'l.) \$99.00

To order or request your free Apple developer tools catalog, call APDA®:

U.S.: 1-800-282-2732

Canada: 1-800-637-0029

International: 716-871-6555

© 1994 Apple Computer, Inc. All rights reserved.

Apple, the Apple logo, APDA, HyperCard, MacApp, Macintosh, MPW, are registered trademarks of Apple Computer, Inc. AppleScript, and QuickTime are trademarks of Apple Computer, Inc.

methods are semantically identical to append and delete operations on lists. One thing to note is that the 'Remove' method gives you the option of disposing of the task or merely removing it from the list. The two 'HowMany???' methods let you count the total number of objects in the queue, or the number of objects belonging to a specific class. The code for `HowManyTasksOfType` is shown below, and demonstrates the use of the `ClassName` method.

```
long CPPTaskManager::HowManyTasksOfType (char *ClassName)
/* returns how many objects of type "ClassName" are in the queue */
{
    long i = 1;
    long Count = 0;

    while (i <= numItems)
    {
        if (strcmp (((*this)[i])>ClassName(), ClassName) == 0)
            Count ++;
        i++;
    }
    return Count;
}
```

The key method in the PTM class is `RunPeriodicTasks`, which iterates through all the tasks which it manages, executing those whose periods have expired and removing those which have completed. The code for this method is shown below:

```
void CPPTaskManager::RunPeriodicTasks (void)
/* Give all of the periodic tasks we manage a chance to execute */
{
    long i = 1;
    CPPPeriodicTask *TheTask;

    while (i <= numItems)
    {
        if (TheTask = (CPPPeriodicTask *)((*this)[i]))
        {
            // execute the task if it needs to run
            if (TheTask->NeedsToRun())
            {
                TheTask->isRunning = TRUE;
                BroadcastMessage (kTaskActivated, (void *)i);
                TheTask->DoPeriodicAction();
                TheTask->isRunning = FALSE;
                BroadcastMessage (kTaskDeactivated, (void *)i);
            }

            // if it has completed remove it from the queue
            if (TheTask->hasCompleted)
            {
                CPPList::DeleteItem(i, FALSE); // Dequeue w/o deleting
                TheTask->DoCompletedAction(); // Perform final operation
                if (TheTask->deleteWhenDone) // Dispose of the task if
                    delete TheTask;        // requested
            }
            else // advance to the next item if we don't delete this one
                i++;
        }
    }
}
```

On each call to `RunPeriodicTasks`, each task in the queue is asked to determine whether or not it needs to run; if it does, the PTM calls `DoPeriodicAction` (the contents of which will be explained in a minute). If during execution, the periodic task determines that it is done, the PTM removes the task from the queue, calls `DoCompletedAction` to let the task perform any special final actions, and then deletes it, if requested to. Ideally `RunPeriodicTasks` should be called once during

every iteration of the application's main event loop in order to give enough processing time to the tasks in the PTM.

`CPPPeriodicTask` is an abstract base class whose only job is to maintain all the bookkeeping information which the PTM needs to service it. Each task has associated with it a minimum period which indicates (in ticks) how often it is to run. Unlike the Time Manager, this is not a guaranteed rate of service – it merely indicates that the task will be run no more often than once every *n* ticks. While this is an acceptable strategy for polling periodic lookup/read/write tasks, I wouldn't advise it for controlling vital processes.

The key functional methods in `CPPPeriodicTask` are `NeedsToRun`, `DoPeriodicAction` and `DoCompletedAction`. The basic `NeedsToRun` method subtracts the last called time of the task from the current time, and returns TRUE if that amount exceeds `minimumPeriod`. If you want you could override it to take the state of the task into account, as well as the period.

The basic `DoPeriodicAction` method merely increments the `timesCalled` variable and stores the current time in `lastCalled`. Any override of `DoPeriodicAction` should call the inherited method so that this bookkeeping information is maintained. An important job which must be done by the user's `DoPeriodicAction` method is setting the public `hasCompleted` flag, which the PTM checks to see whether the task should be dequeued.

`DoCompletedAction` allows the user to customize the task's final behavior in two ways. Here is the code for the basic method:

```
void CPPPeriodicTask::DoCompletedAction (void)
/* execute the completion routine, if there is one */
{
    if (this->completion)
        (*this->completion)((CPPObject *)this);
}
```

One way to customize `DoCompletedAction` would be to overload it to perform any necessary data copying, notification, etc. The other way is to use `SetCompletionProc` to specify a routine to be called explicitly. Why do we need to be able to do it both ways? Glad you asked! As an example, most of the ACL's lookup tasks allocate goodly-sized chunks of memory which are only used when a lookup is underway. Principles of information hiding make it more logical to dispose of this memory in an overridden method than an external routine.

On the other hand, many times when we do a lookup we would like the results to be displayed somewhere. Principles of information hiding argue that this kind of 'display-specific' code should not be placed inside the task itself. In addition, it would become tedious and wasteful to have to create several subclasses of the same kind of lookup task which differ only in their completion routines. This all becomes much clearer when we discuss the construction of the Yenta application, so don't worry if this doesn't quite make sense yet.

FREE INSTALLER!

To: Mac Developers & Product Managers

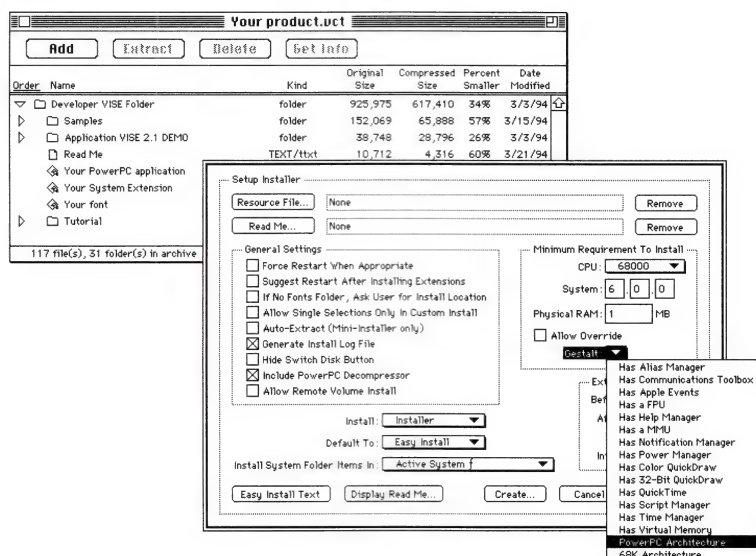
From: MindVision (Creators of Stacker for Macintosh)

Subject: Free Copy of DEVELOPER VISE 3.0

**Message: We've got a great new installer for you.
Here's your chance to try it for free.
No risk, no obligation, no hassle.
Call today, don't delay!**

**Full PowerPC
Support**

**Integrated
compression**



**No programming
required**

**Fully graphical
interface**

Call (402) 477-3269

IF YOU PREFER, USE E-MAIL. APPLELINK, AOL: MINDVISION • COMPUERVE 70253,1437

Join the growing list of companies who use our VISE technology: Adobe, Apple, CE Software, Claris, CompuServe, DeltaPoint, MacroMedia, Radius, Stac Electronics, Symantec, WordPerfect, and many more.



MindVision Software 840 South 30th St., Suite C, Lincoln, Nebraska 68510
Voice: (402) 477-3269 Fax: (402) 477-1395 AppleLink, AOL: MindVision

© 1992-94 MindVision Software. All Rights Reserved. Developer VISE is a trademark of MindVision Software.





LS Object Pascal



Powerful ■ Faster ■ Native
shipping on CD-ROM for PowerMac
now \$399 was ~~\$695~~

1-800-252-6479

Language Systems Corp. • 100 Carpenter Dr. • Sterling, VA 20164 • (703) 478-0181 • Fax: (703) 689-9593 • AppleLink: LANGSYS

PORING (OVER) THE CONCRETE

Now that we have discussed the abstract classes, let's move on to the ones which do the gruntwork.

MaBell

In order to use Appletalk, you need a set of maintenance routines which let you do things like find out driver numbers, turn self-send capability on and off, initialize the PPC Toolbox, etc. To save the programmer worry (and time spent flipping through IM) I encapsulated these routines in a class called CPPMaBell, whose declaration is shown below

```
class CPPMaBell : public CPPObject {
public:
    CPPMaBell (Boolean AllowSelfSend = TRUE);
    ~CPPMaBell (void);

    virtual char *ClassName (void);

    // General Appletalk Utilities
    OSErr    OpenATalkDriver (short whichDrivers);
    short    GetDriverNumber (short whichDriver);
    short    GetAtalkVersion (void);
    Boolean  HasPhase2ATalk (void);
    Boolean  EnableSelfSend (void);
    Boolean  DisableSelfSend (void);
    Boolean  SelfSendEnabled (void);

    // PPC Toolbox routines
    OSErr    InitPPCToolbox (void);
```

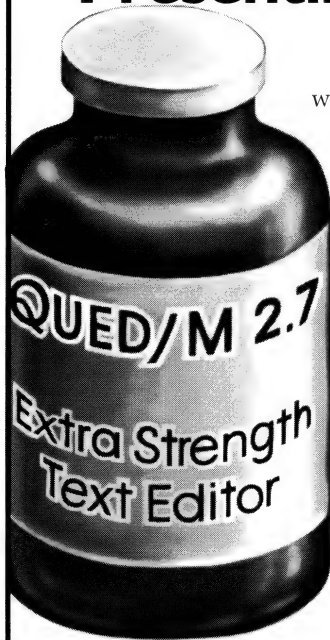
```
OSErr    OpenCommunicationPort (StringPtr OurName,
                                StringPtr PortType,
                                PPCPortRefNum *newRefNum);
OSErr    CloseCommunicationPort (PPCPortRefNum portRefNum);
OSErr    EndSession (PPCSessRefNum sessionID);
private:
    short    XPPRefNum,
            MPPRefNum,
            ADSPRefNum;
    short    AtalkVersionNumber;
    Boolean  SelfSendState;
};
```

The majority of these routines are unspectacular, but necessary. One routine of interest is `OpenCommunicationPort`, which shows how to open a PPC Toolbox Port. This code modifies the example code in IM:IAC, p. 11-21, by setting the connection's full address to "OurName:PPCCommPort" at location "?s Macintosh: OurName•PortType@ zone".

```
OSErr CPPMaBell::OpenCommunicationPort (StringPtr OurName,
                                         StringPtr PortType,
                                         PPCPortRefNum *newRefNum)
{
    PPCOpenPBRec    OpenRec;
    PPCPortRec       PortRec;
    LocationNameRec  LocationRec;
    EntityName       TheName;
    OSErr            ErrCode;

    // set up the port specification record
    PortRec.nameScript = smRoman;
    PortRec.portKindSelector = ppcByString;
    CopyString (OurName, PortRec.name);
```


Presenting the extra-strength text editor.



If you thought previous versions of QUED/M were powerful, wait until you program with QUED/M 2.7, loaded with these new pain-relieving features:

- ⇒ Integrated support for THINK Project Manager™ 6.0 & 7.0
- ⇒ THINK™ debugger support
- ⇒ CodeWarrior™ support (now it's easier than ever to program for the Power Macintosh®!)
- ⇒ MPW ToolServer™ support
- ⇒ PopUpFuncs™ support
- ⇒ Frontier™ Do Script support

**For quick relief, call
800-309-0355 today.**

Of course, QUED/M 2.7 still has all the features that make it easier to use than any other text editor:

- ⇒ Macro Language lets you automate tedious tasks
- ⇒ Search and Replace through multiple unopened files and using GREP metacharacters
- ⇒ File comparisons using GNU Diff
- ⇒ Unlimited undos and redos
- ⇒ 10 Clipboards that can be edited, saved, and printed
- ⇒ Customizable menu keys
- ⇒ Text folding
- ⇒ Display text as ASCII codes
- ⇒ Plus many more features!

Get even more relief: Try QUED/M 2.7 now for just \$69! You'll get a 30-day money back guarantee too! Call now to order. 800-309-0355.

QUED/M is a trademark of Paragon Concepts, Inc. All other products are trademarks or registered trademarks of their respective holders.

107 S. Cedros Ave. • Solana Beach, CA 92075 • Tel (619) 481-1477 • Fax (619) 481-6154

NISUS
Software Inc.

```
CopyString ("\pPPCCommPort", PortRec.u.portTypeStr);
```

```
// set up the record which describes who we are
LocationRec.locationKindSelector = ppcNBPTTypeLocation;
PStrCat(32, &LocationRec.u.nbpType[0], 3, OurName,
        "\p.", PortType);
```

```
// set up the 'open port' record
OpenRec.serviceType = ppcServiceRealTime;
OpenRec.resFlag = 0;
OpenRec.portName = &PortRec;
OpenRec.locationName = &LocationRec;
OpenRec.networkVisible = TRUE;
```

```
// make the call and, if no errors, get the port number
if ((ErrCode = PPCOpen (&OpenRec, FALSE)) == noErr)
    *newRefNum = OpenRec.portRefNum;
else
    *newRefNum = -1;
return ErrCode;
}
```

CPPNODEINFO

As mentioned in the Appletalk Review, network entities can be identified using three strings and/or three numbers. The CPPNodeInfo object whose declaration is shown below is the ACL's common denominator for representing network entities. All six pieces of address data are freely settable and accessible, placing no restrictions on the protocol you use to communicate with (eg. DDP uses only the numbers, PPC only the names).

```
class CPPNodeInfo : CPPObject {
public:
```

```
    CPPNodeInfo (void);
    ~CPPNodeInfo (void);
    virtual    char *ClassName (void);

    void    SetNodeName (StringPtr ObjectStr, StringPtr TypeStr,
                        StringPtr ZoneStr);
    void    SetNodeAddress (short SocketNum, short NodeNum,
                        short ZoneNum);
    void    GetNodeName (StringPtr *ObjectStr, StringPtr *TypeStr,
                        StringPtr *ZoneStr);
    void    GetNodeAddress (short *SocketNum, short *NodeNum,
                        short *ZoneNum);
    StringPtr    ReturnNameString (void);
    StringPtr    ReturnShortNameString (void);

    Boolean    RegisterNodeAddress (OSErr *ErrCode);
    Boolean    DeregisterNodeAddress (OSErr *ErrCode);
    Boolean    ConfirmNodeAddress (OSErr *ErrCode);

    Boolean    Equals (CPPNodeInfo *TestNA);
    CPPObject *Clone (void);
    Ptr    InfoToStream (void);
private:
    NamesTableEntry *NTE; // only defined if registered
    StringPtr    objectString,
                typeString,
                zoneString;
    unsigned char socketNumber,
                nodeNumber;
    short    zoneNumber;
    Boolean    isRegistered;
};
CPPNodeInfo *StreamToInfo (Ptr Buffer, Ptr *BufferEnd);
```

In addition to managing the network address and name data, CPPNodeInfo lets you register and deregister address on the

**Power Mac and
Macintosh
Developers:**

FIND OUT FAST WHAT'S GOING ON IN MEMORY

THE MEMORY MINE™

- See memory allocation in any open heap at a glance.
- Easily spot memory leaks.
- Flags heap corruption when it happens.
- Works with source level debugger to let you find memory problems fast.
- Stress applications on the fly with Purge, Compact, and Zap.
- Allocate memory at will for precise stress testing.
- Log heap data - easily document heap status over time.
- No need for source code: nothing inserted in code; no patches to the system.
- Works with 24-bit, 32-bit, and modern memory managers.

For Macintoshes with 68020 or better. Requires System 7.0 or later.

only \$99 US
Order now from Adianta, Inc.


Phone: (408)354-9569 • FAX: (408)354-4292

AppleLink: ADIANTA • AOL: Adianta • Internet: adianta@aol.com

For VISA, MC, or American Express orders by mail, fax, or Applelink, please include name, address, card number, expiration date, and phone number or email address.

Also available through the MacTech Mail Order Store.

for more information contact

 **Adianta, Inc.** • 2 N. Santa Cruz Ave. #201 • Los Gatos, CA 95030

network, so that any CPPNodeInfo entity can be made visible to the outside world. For convenience, a synchronous routine, ConfirmNodeAddress, has been provided to let you check to see if the entity's address has changed behind your back.

InfoToStream and StreamToInfo provide the ability to write the node's address out to a block of memory and then reconstruct a CPPNodeInfo record from such a block of memory. This is provided so that you can send addresses back and forth between machines without any loss of data.

Finally, CPPNodeInfo provides a Clone method which creates a CPPNodeInfo object which refers to the same object as the original, and an Equals method, which tests a passed-in object to see if it refers to the same object as the one which makes the method call.

411 – CPPPERIODICTASK RETURNS

In the Appletalk Review, we talked about two different kinds of lookup we want to be able to do – lookup of zone names (including the zone we are in) and lookup of all network entities whose names match a predefined pattern. In the ACL, these two jobs are performed by subclasses of CPPPeriodicTask, affectionately known as CPPZone411 and CPPNode411. Here are their declarations (for brevity, I've omitted their private variable and method declarations).

```
class CPPZone411 : public CPPPeriodicTask {
public:
    CPPZone411 (CPPTaskManager *TaskManager,
                CPPMaBell *MaBell, long minPeriod = 120,
                Boolean deleteWhenDone = TRUE);
    ~CPPZone411 (void);
    virtual char *ClassName (void);

    virtual void DoPeriodicAction (void);
    virtual void DoCompletedAction (void);
    Boolean NthZone (long whichItem, Boolean getCopy,
                    StringPtr *ZoneName);
    long NumZonesFound (Boolean *isDone);

    void StartZoneLookup (CompletionProc DoProc);

    StringPtr GetOurZoneName (OSErr *ErrCode);
    CPPStringList *GetFoundList (void);

protected:
    CPPStringList *FoundList;
private:
    ...
};

class CPPNode411 : public CPPPeriodicTask {
public:
    CPPNode411 (CPPTaskManager *TaskManager,
                long minPeriod = 120,
                Boolean deleteWhenDone = TRUE);
    ~CPPNode411 (void);
    virtual char *ClassName (void);

    virtual void DoPeriodicAction (void);
    virtual void DoCompletedAction (void);

    CPPNodeInfo *NthNode (long whichItem, Boolean getCopy);
    long NumNodesFound (Boolean *isDone);
    void StartNodeLookup (StringPtr ObjectName,
                        StringPtr TypeName,
                        StringPtr ZoneName,
                        short maxNumResponses,
                        CompletionProc DoProc);
    CPPObjectList *GetFoundList (void);

protected:
    CPPObjectList *FoundList;
private:
    ...
};
```

Each class maintains a list of names or nodes which it has found, and provides a similar interface (Nth???, Num???Found, and GetFoundList) for accessing them. CPPZone411 provides an additional routine called GetOurZoneName which returns the name of the network our machine resides on. The details of how to perform zone and node lookup have been discussed many times in IM and programs like DMZ and GetZoneList from Apple DTS, so I won't go into those here. It is, however, worth looking at the DoPeriodicAction, DoCompletedAction, and Start???Lookup methods to see how you can use the CPPPeriodicTask abstract class to do something useful. We'll look at CPPNode411, since node lookup is considerably more straightforward.

To begin a node lookup, one calls the StartNodeLookup method, passing it the name, type, and zone names to match against, along with the maximum number of responses you want, and the address of a routine to call on completion. Here are the guts of StartNodeLookup.

Cross-Platform Object Database Engine

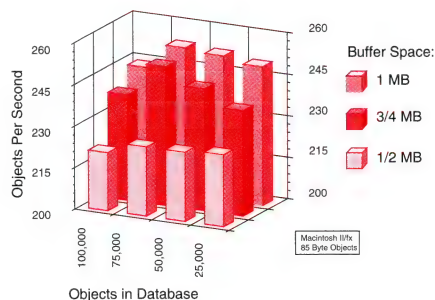
Full Featured

NeoAccess™ allows you to embed the power of a full-featured object-oriented database engine into your **Macintosh, Windows** and **DOS** based applications. No other object database engine can match NeoAccess's impressive feature set, including: Blobs, part lists, iterators, swizzlers, temporary objects, multiple indices on a class, schema evolution, a powerful relational query mechanism, a streams-based i/o model and incredible performance.

High Performance

Internally, NeoAccess uses extended binary trees and binary search algorithms to achieve optimally short access times. Its automatic query optimizer ensures that queries always use the fastest access path to objects. And indices are dynamically combined, collapsed and compressed to keep access times to an absolute minimum as the contents of a database changes. NeoAccess's object caching boosts performance by keeping objects in memory even after being disposed of by the application. Your application's memory size is reduced because only those objects of immediate interest need to be in memory at any one time, not the entire file.

Random Access



Full Source Code

We've taken a frameworks approach toward object persistence and database technology. In much the same way that application frameworks are used to construct the front-end of an application, NeoAccess is the framework you use to build your application's back-end. As is the case with virtually all object frameworks, the NeoAccess Developer's Toolkit comes complete with **full source code**, for all major application frameworks including Metrowerks' **PowerPlant**, Symantec's **THINK Class Library** and Apple's **MacApp** on the Macintosh and **Microsoft's Foundation Classes**, Inmark's **zApp** and Borland's **ObjectWindows** in Intel-based environments. It can even be used without a framework or in one that you've designed.

Easy to Use

The programming interface is designed around the concept of minimum visible complexity. Application-specific objects inherit persistence properties from a NeoAccess base class. These objects are organized in the database primarily by class. But NeoAccess also knows how classes are related. So multiple classes can be searched in a single operation. And of course objects of any particular class can be organized using multiple indices. NeoAccess is unique in that it allows objects to be located based on abstract selection criteria or based on their relationship to other objects. There's literally no database administration to deal with – NeoAccess takes care of all the details. NeoAccess also includes a Blob mechanism which allows free-form variable-length data to be stored in databases with the same ease as fixed-length objects. NeoAccess even includes a powerful set of keyed iterators for traversing indices and part lists. Keyed iterators have the unique ability to iterate over only those objects in a set that match a given selection criteria. Your users will appreciate NeoAccess because databases are completely self-contained in a single document file. So users can treat a database file as they would any other document.

Proven

NeoAccess has been commercially available since May of 1992. Hundreds of commercial and in-house applications based on NeoAccess technology have already been deployed. NeoAccess can help your organization deliver powerful products in a more timely fashion than you ever imagined possible.

Affordable

NeoAccess's best feature is its price. The NeoAccess Developer's Toolkit sells for just \$749 per developer with absolutely **no runtime licensing fees**. It includes **full source code**, numerous sample applications, 450+ pages of documentation, and 30 days of technical support. So what are you waiting for?

Now on the CodeWarrior CD!
See the NeoLogic folder on the root of the CD-ROM for more details.

neo•logic

In order to survive, you need to persist.

No Runtime Licensing Fees!
Use NeoAccess in all your development for one low price!

Databases in the Finder?

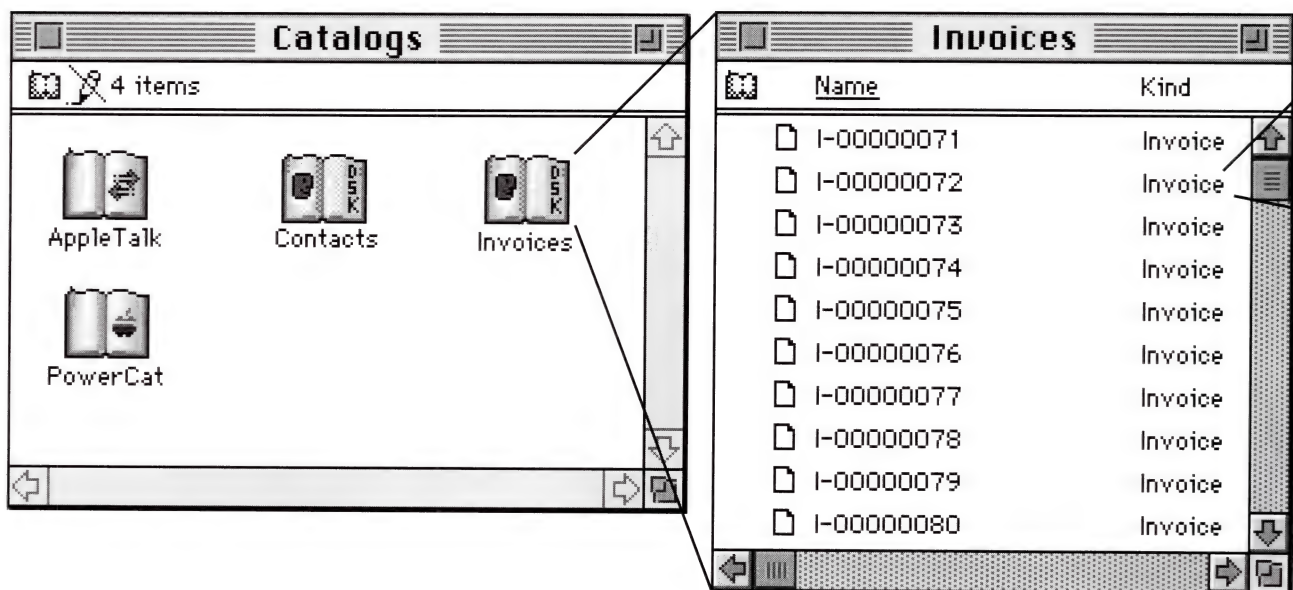
Unfortunately, most users must interact with relational databases at the SQL level or behind a front-end that can take months to develop.

```
select invoice_num,invoice_date,cust_id,cust_name from invoices,customers where invoice_id = 112;printall;
```

Why not view and edit your data at the Finder level? Announcing **Cataloger™**.



Cataloger brings a whole new way of working with databases to the Macintosh. The Finder™ has always provided a view of our desktop files, why not database records too?



Apple's **Open Collaborative Environment** places a catalog icon at the Finder. Open the icon up and you'll see a window like the one above. The AppleTalk catalog shows nodes on the network. The PowerCat catalog shows users and groups in a PowerShare Server.



The Contact and Invoices catalogs (for example) can show information from **4D Server**, **DAL/DAM**, **Oracle** and **Sybase** databases! If your needs are more customized, you can write AppleScript handlers to provide the contents of your catalog.

The Catalog Manager (an AOCE API) allows for **Catalog Service Access Modules** (CSAMs) to be created that interact with external sources.

The Finder and other applications make requests through the Catalog Manager and Cataloger does the rest. There are no limits to the amount of data brought back.





Templates provide the ability to view and edit individual records. AOCE Templates are designed with **Template Constructor™**. This powerful application let's you easily design forms to view data coming from several sources, control behavior with C and AppleScript, and view row/column data with **TableIt!™** - a powerful data display tool.

I-00000072

Invoice #: **I-00000072** Invoice Date: **9/1/94**

Company: **Tall Timbers Marina** Terms: **Net 30 Days ▼**

Qty	Part #	Description	Cost Each	Line Total
1	2143	Propeller	250	250
2	3853	Drain Plugs	12.90	25.80
1	4905	Main Light assembly	150	150
8	9384	Spark Plugs	4.00	32.00
1	9999	Labor	600	600

Notes: Prepared boat for summer season.
Dewinterized, tuned up and replaced main propeller. Light assembly replaced after testing front lights failed.

Subtotal: **1057.80**
Tax: **52.89**
Total: **1110.69**



Access information and catalog definitions are stored in and protected by the AOCE **Keychain**. Your users can have one username and password for a variety of data sources!



Need more horsepower? **Database Scripting Kit™** provides AppleScript access to all of the connection information and data sources. Write AppleScript routines that send queries and parse results from within your own application.

Cataloger/Template Constructor includes:

Database Scripting Kit™ with connectors for 4D Server, DAL/DAM, Oracle, Sybase and others.

DSK Cataloger™ CSAM

Template Constructor™ with TableIt!™

Several example catalogs/templates and over 100 example AppleScripts.

Complete scriptability, native for Power Macintosh. Drag Manager and Thread Manager support.

How to contact us:

**Graphical Business
Interfaces, Inc.**

Voice: 800-424-7714 or 219-253-8623

Fax: 219-253-7158

E-Mail: steve@gbi.com

GBI. Bringing your data into view.™

**Now
Shipping!**

MacForth Plus & Power MacForth



The Language of Innovation is now available native on the Power Macintosh.

FEATURES

- Royalty Free Turnkey Compiler
- Text Editor
- Online Glossary Tool
- System 7 Compatibility
- 68020 Assembler
- 68881/2 Co-processor Support
- High Level Graphics
- Toolbox Support
- Extensive Documentation
- Reasonable Upgrades
- Optional Tools Disks
- MacForth Plus \$199
- Power MacForth \$199



Creative Solutions, Inc.

4701 Randolph Road, Suite 12
Rockville, MD 20852
301-984-0262 or 1-800-FORTH-OK (orders)
Fax: 301-770-1675 Applelink: CSI

```
void CPPNode411::StartNodeLookup (StringPtr ObjectName,
    StringPtr TypeName,
    StringPtr ZoneName,
    short maxNumResponses,
    CompletionProc DoProc)
{
    if ((!this->hasCompleted) || (!this->ourManager))
        return; // exit if doing a lookup or bogus params

    // otherwise, set up the call
    this->SetCompletionProc(DoProc);
    this->hasCompleted = FALSE; // note we are not done
    this->FoundList->DeleteItems(TRUE); // get rid of old results

    SetupNBPCall(ObjectName, TypeName, ZoneName, maxNumResponses);

    if (this->callResult == noErr)
        this->ourManager->AddPeriodicTask(this); // now add the task
}
```

The first thing the task does is check its own `hasCompleted` flag to see if it is currently doing a lookup. Once it determines that it is not already busy, it sets the flag to `FALSE`, so that the PTM will not toss it out as soon as it is enqueued. It then clears any nodes which may have been found earlier from its list of 'found' nodes. `SetupNBPCall` is a private method which allocates the necessary storage, fills in the `paramBlock` for an NBP call, and makes the `PLookupName` call – storing its result in the class' `callResult` variable. If the call is made successfully, the only thing left to do is add the task to the PTM's list of serviceable tasks. All subsequent activity is managed by the `DoPeriodic/CompletedTask` methods.

Here's `CPPNode411`'s `DoPeriodicAction` method:

```
void CPPNode411::DoPeriodicAction (void)
{
    // call the inherited method to update frequency count
    CPPPeriodicTask::DoPeriodicAction();

    switch (this->lookupRec->MPPioResult) {
        case noErr : // the call has completed
            this->ProcessNodes(lookupRec->NBPnumGotten);
            this->hasCompleted = TRUE;
            this->callResult = noErr;
            break;

        case 1 : // still busy getting names
            break;

        default : // an error occurred
            this->callResult = lookupRec->MPPioResult;
            hasCompleted = TRUE;
            break;
    }
}
```

As one might expect, the periodic task checks the `ioResult` parameter of the `paramBlock` which was used to make the `PLookupName` call and responds appropriately. If it has completed successfully, it calls `ProcessNodes` which extracts all of the nodes from the lookup buffer, then sets the `hasCompleted` flag so the PTM will remove it from the queue. If the call has completed with an error, the task also sets the `hasCompleted` flag, but records the error so that the programmer can use the `TaskError` method to find out what went wrong.

When `RunPeriodicTasks` sees that the `hasCompleted` flag is set, it will call `CPPNode411`'s `DoCompletedAction` method:

```
void CPPNode411::DoCompletedAction (void)
{
    NukePtr(this->returnBuffer);
    NukePtr(this->lookupRec);
    inherited::DoCompletedAction();
}
```

As mentioned earlier, this method is used exclusively to free up memory allocated in `StartNodeLookup` before calling the inherited method which will call the completion routine the user passed to `StartNodeLookup`. Here's an example of such a completion routine:

```
void NodeLookupCompleted (CPPObject *TheTask)
/* add each 'found' node to a list of users */
{
    CPPNode411 *LookupTask = (CPPNode411 *)TheTask;
    Boolean BTemp;
    long numFound;

    numFound = LookupTask->NumNodesFound(&BTemp);
    if (numFound && BTemp)
    {
        for (long i = 1; i <= numFound; i++)
            UserList->AddNode(LookupTask->NthNode (i, FALSE));
    }
}
```

Though the guts of the `Start`, `DoPeriodic`, and `DoCompleted` code are different in `CPPZone411` and every other periodic task, the basic strategy remains the same: allocate memory for the call in `Start`, check the `ioResult` flag in `DoPeriodicAction`, and deallocate the memory in

The database engine for professionals

CXBase Pro

Based on the database engine used by **Apple AOS** for their **eWorld** content publishing software, *CXBase Pro* represents the state of the art in data storage and retrieval technology. *CXBase Pro* has all the tools you need to develop any kind of application that needs its own database engine.

The philosophy behind *CXBase Pro* is to provide a simple, consistent, and powerful API that lets you work in whatever way you choose. *CXBase Pro* is delivered with full source code, giving you maximum flexibility and security. The *CXBase Pro* library provides both multi-keyed data record handling, and flexible BLOB storage where you define the data format. And *CXBase Pro* has been engineered from the ground up for maximum performance.

Test drive before you buy - Full-featured \$49 demo available!

CXBase Pro also comes in a demo version. You get the complete manual, and a full-featured library limited only by file size. You get access to all the function calls, and you can use it as long as you like. If you decide to buy the full package you get credit for the price of the demo.

For full prices, to order, or to receive more information, contact:

TSE International | Taandwaarsstr. 51 | 1013 BV Amsterdam, Holland | tel: 31 20 638-6507 | fax: 31 20 620-4933 | AppleLink : **TSE.INT**

Apple, AOS, eWorld, Power Macintosh, and Quadra are trademarks of Apple Computer, Inc.

■ Powerful and flexible select module

CXBase Pro lets you create multiple kinds of selections, and provides a callback mechanism that allows you to both display the progress, and capture the data as it is selected.

■ Native PowerPC version

What better way to use the speed of a Power Macintosh than to rev up your data access? Average native speed 3 times that of the fastest Quadra.

■ 100% Cross-platform

Both the source and data files are compatible across all platforms. Yes, Virginia, it really works!

■ Access to logical and absolute record numbers

You can access data records not only by key, but also by logical and absolute record numbers.

■ Royalty-free license

You pay once for the development license. Period!

DoCompletedAction.

A third task which falls partially under the pervue of 411 is `CPPConfirmTask`, which takes a `CPPNodeInfo` object and a completion routine, and sets a flag to indicate whether the node still exists on the network. The unique parts of its public declaration are shown below:

```
class CPPConfirmTask : public CPPPeriodicTask {
public:
    CPPNodeInfo*NodeToConfirm;
    CPPConfirmTask (CPPTaskManager *TaskManager,
                    long minPeriod = 120,
                    Boolean deleteWhenDone = TRUE);
    ~CPPConfirmTask (void);
...
    Boolean NodeExists (void);
    void StartNodeConfirm (CPPNodeInfo *TheNode,
                           CompletionProc DoProc);
private:
...
};
```

THE THREE R'S...AND THE OTHER GUY

When using the PPC Toolbox to exchange data, the strategy is to open a connection on your computer, wait for someone to start a session with you, perform all necessary reads and write, then close the session. We will, therefore, need four more tasks to let us exchange data over the network - 'read', 'write', and 'respond to connection requests'. Oh yes, and 'initiate connection requests'.

Responding...

Listening for a connection is done by posting a `PPCInform` call; connections can be accepted or rejected based on authentication information which the caller provides. In the ACL, `CPPMaBell`'s `OpenCommunicationPort` is used to open a port on your machine which someone can connect to. A subclass of `CPPPeriodicTask` called `CPPListenTask` does the 'waiting' by posting and polling a `PPCInform` call. In its current form, it automatically accepts all connections which people try to form with it. If you wanted to provide authentication, you could simply subclass `CPPListenTask`, set it to not automatically accept, then have its `DoCompletedAction` method perform the needed verification. The unique parts of the declaration for `CPPListenTask` are shown below:

```
class CPPListenTask : public CPPPeriodicTask {
public:
    CPPListenTask (CPPTaskManager *TaskManager,
                    long minPeriod = 120,
                    Boolean deleteWhenDone = TRUE);
    ~CPPListenTask (void);
...
    PPCSessRefNum GetNewSessionID (void);
    void GetConnectedToInfo (Boolean *hasConnected,
                             LocationNamePtr *Location,
                             PPCPortPtr *PortRec);
...
};
```



```

void      StartListenTask (PPCPortRefNum PortID,
                          CompletionProc DoProc);
protected:
    PPCSessRefNum sessionID;
private:
    ...
};

```

You start the connection task by passing it the reference number of the port opened by CPPMaBell, and a completion routine. When the task completes, you can get the reference number for the session, and information about the person you are connected to using the GetNewSessionID and GetConnectedToInfo methods. The guts of StartListenTask are merely a modification of the code in *IM: IAC*, p. 11-36, and so are not repeated here.

Reading...

Once a connection is open, a PPCRead task should be posted asynchronously to receive any data coming in on the connection. The ACL task which lets you do this is called CPPReadTask; the unique parts of its declaration are shown below.

```

class CPPReadTask : public CPPPeriodicTask {
public:
    CPPReadTask (CPPTaskManager *TaskManager,
                 long minPeriod = 120,
                 Boolean deleteWhenDone = TRUE);
    ~CPPReadTask (void);
    ...
    long      GetDataSize (void);
    Handle    GetData (Boolean BecomeOwner,
                      Boolean *AmITheOwner);
    void      StartReadTask (PPCSessRefNum ConnectionID,
                            short blockSize,
                            CompletionProc DoProc);
protected:
    PPCSessRefNum sessionID;
private:
    ...
};

```

You start a CPPReadTask by specifying a session number (obtained from CPPListenTask) and the size of the blocks you want to read. CPPReadTask starts out with an empty data handle, and a buffer of size blockSize in which data is temporarily stored. As data is sent over to it from the other end of the connection, it is accumulated into the data handle in blockSize sized chunks. The 'optimal' block size to use will depend heavily on the application; for transmitting typed messages, 128 bytes is probably fine; for doing file-transfers, you probably want to go to 1024 or 2048 bytes per block.

When the task is completed, GetDataSize tells you how much was read, and GetData returns a handle to the data and lets you establish ownership of it. The distinction of who owns the actual data handle becomes important when more than one object calls GetData on the same CPPReadTask. It is also important to determine whether the data will remain after the CPPReadTask is deleted. When you call GetData, you use BecomeOwner to tell it whether or not you will take responsibility for disposing of the handle. When you get the handle back, the AmITheOwner variable is set to indicate

whether you succeeded in getting ownership of the handle, or whether someone else already owns it.

Note that a good time to post the first CPPReadTask on an open connection is in the completion task of the CPPListenTask which received the connection in the first place. I did not do this in CPPListenTask both to make it more flexible and to reduce the coupling between classes in the ACL, but the CPPYListenTask subclass used by the Yenta application uses this strategy most effectively.

Writing...

On the other end of the connection from PCCInform and PPCRead, a PPCWrite task must be posted asynchronously to dump a chunk of data across the network. This duty is carried out by CPPWriteTask class (declaration below).

```

class CPPWriteTask : CPPPeriodicTask {
public:
    CPPWriteTask (CPPTaskManager *TaskManager,
                 long minPeriod = 120,
                 Boolean deleteWhenDone = TRUE);
    ~CPPWriteTask (void);
    ...
    void StartWriteTask (Ptr DataToWrite, Boolean OwnsData,
                        PPCSessRefNum ConnectionID,
                        CompletionProc DoProc,
                        OSType DataType = "???",
                        OSType DataCreator = "????");
    void StartWriteTask (PPCPortRefNum SourcePortRefNum,
                        CPPNodeInfo *SendTo,
                        Ptr DataToWrite, Boolean OwnsData,
                        CompletionProc DoProc,
                        OSType DataType = "???",
                        OSType DataCreator = "????");
private:
    ...
};

```

To provide flexibility, this task may be started in one of two ways. The first is to pass it a CPPNodeInfo object corresponding to the network entity you want to communicate with, the data, the ownership flag, and the block's type and creator. In this case the write task synchronously establishes the session for you, making the assumption that the person you are trying to talk to has followed the naming conventions for ACL objects (see the description of CPPMaBell). A drawback of using synchronous connect is that you cannot connect to another port on your own computer.

The second way to start CPPWriteTask is to pass it the reference number of an established connection (which requires that you establish the connection yourself) along with the data you want to send, a boolean flag indicating whether it is allowed to dispose of the data on completion, and the type and creator of the data block.

"But how do I establish a connection?" Funny you should ask. The ACL provides a class which lets you asynchronously establish a connection with another network entity and return the reference number of the established connection. The declaration for this class (CPPConnectTask) is shown below.

```

class CPPConnectTask : CPPPeriodicTask {
public:
    CPPConnectTask (CPPTaskManager *TaskManager,

```


EHelp

CALL

(800) 247-7890

OR

(919) 481-3517

FOR A

FREE EHELP

DEMO DISKETTE.

FSI
FOUNDATION
SOLUTIONS INC



Add Multimedia Documentation To Your Applications And Now...

Build Stand-Alone Documentation With EHelp 4.0

THE PREMIER DOCUMENT DISPLAY SYSTEM FOR THE MACINTOSH

- Integrated multimedia elements
- Flexible hot-spot linking
- Robust hypertext capabilities
- Sophisticated searching
- Easy integration with your applications
- Reads WinHelp™ and MSWord™ source
- Apple-event aware
- Function macros for inter-document jumps, control of user buttons, launching other applications, etc.
- World Ready for all Macintosh-supported languages
- Many other features

FAX (919) 481-3551

EHelp 4.0 is a trademark of Foundation Solutions, Inc.
EHelp 3.0 is still available and supported.

```

        long minPeriod = 120.
        Boolean deleteWhenDone = TRUE);
~CPPConnectTask (void);

...
PPCSessRefNum GetSessionID (Boolean *isDone);
void          StartConnectTask (
                PPCPortRefNum SourcePortRefNum,
                CPPNodeInfo *ConnectTo,
                CompletionProc DoProc);

protected:
    PPCSessRefNum sessionID;
private:
    ...
};

```

As with the second StartWriteTask call, all you have to do is pass it the reference number of the port on your computer, and the network address of an entity which you wish to talk to and has followed the naming conventions for ACL network objects. When the task completes, you can use the GetSessionID call to get the reference number to pass to the second StartWriteTask call.

And...Closing?

When you have finished reading and writing data across a session, you can call CPPMaBell::CloseSession to close the connection between the two computers. Something to be wary of when you are figuring out where to make the call to CloseSession, is that closing from the writer's end can sometimes interrupt the read task. It is much better to have the

end which does the reading close the session when it completes so that you can be sure that it actually receives all the data it was sent. Ideally CloseSession would be called by CPPReadTask's completion method, which could then post another CPPListenTask (sort of like having CPPListenTask's completion method posting a CPPYReadTask). The ACL's CPPReadTask does not do this, since it would create restricting dependencies on the CPPReadTask and CPPListenTask, but you are encouraged to implement this behavior in a subclass of CPPReadTask.

THAT'S ALL FOLKS

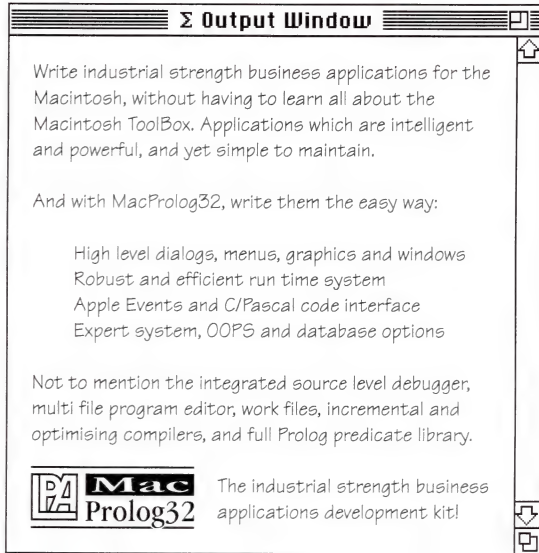
Believe it or not, we've covered the entire Appletalk Class Library. As you've noticed, the only data transmission protocol I use is that provided by the PPC Toolbox. With the information provided by the CPPNodeInfo class, you should be able to subclass CPPPeriodicTask to provide support for ATP, DDP, ADSP, or any other protocol you care to use.

YENTA

Yenta, as I mentioned before, is an application built entirely in C++ which uses the ACL to do its communications, and a set of home-rolled TCL-like classes to provide the interface. The main window of the application is shown in Figure 1.

Industrial Strength

File Edit Search Windows Fonts Eval



Logic Programming Associates Ltd
Phone (US Toll Free): 1-800-949-7567
Phone: +44 81 871 2016 - Fax: +44 81 874 0449
Email: lpa@cix.compulink.co.uk - Applelink: UK0049

they are placed in the list on the right side, which tracks the names and addresses of all users who you know about on all of the zones listed on the left.

Near the bottom is a scrollable text area in which you can type up to 32K of text (can we say TEHandle? I knew you could). To send the text to another user(s), you can either double-click on a single name in the user list, or shift-click on several names and press the send button. If the 'echo' checkbox is on, your message is placed in a scrolling text-edit window (shown in figure 2) along with a log of all your incoming messages.

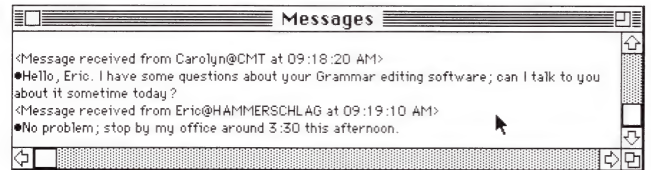


Figure 2. The Yenta "Message Log" Window

In case you leave your computer for any length of time, a feature called AutoReply in the file menu lets you type in a string which is echoed back to any other Yenta application which sends you a message when AutoReply is active.

Also available from the File menu is a Preferences dialog (shown in figure 3) which lets you give Yenta explicit instructions on how to maintain the user list. By default, users are added to the list when they send you messages or whenever you ask Yenta to scan a zone for new users. In the Preferences dialog you can specify a rate at which the application should 1) scan all known zones for new users, and 2) confirm each of the addresses in the right-hand list. Scan causes new users to be added to the list automatically, and confirm causes them to be removed if they are no longer visible on the network. You can also use the Preferences dialog to specify a sound to be played when messages are received (very useful!) and when a user logs on or logs off (marginally useful).

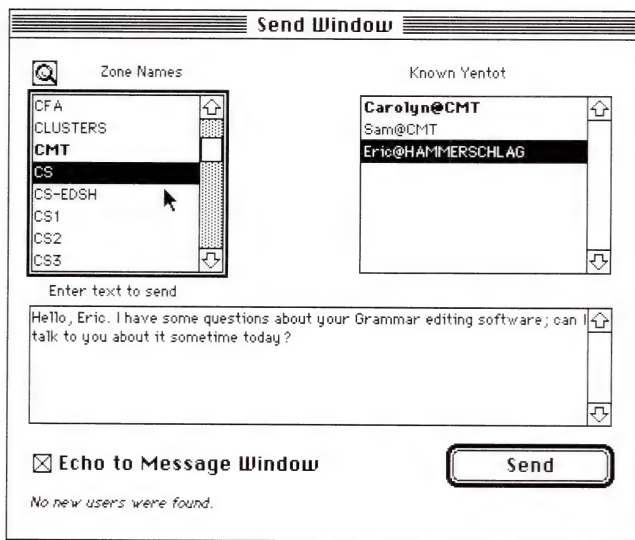


Figure 1 - the Yenta "Send" Window

On the left is a list of all known zones. The magnifying glass button will re-load the list of zones which is automatically loaded when you run the application. Double-clicking on an item in the zone list will cause the program to scan that zone for other Yenta applications. If any are found,

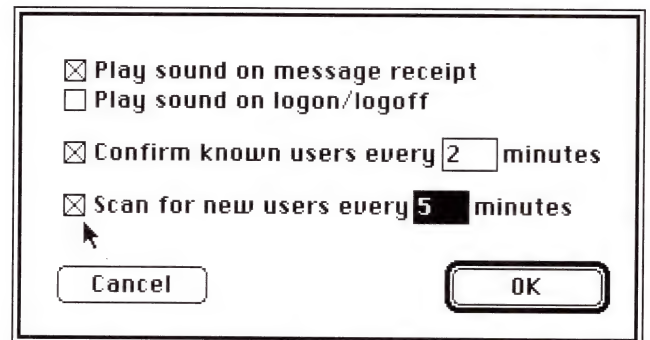


Figure 3. The Yenta "Preferences" Dialog

REQUIREMENTS

Yenta is set to run in a 500 K partition, and should function properly on any Macintosh running System 7 with program linking turned on.

WHAT'S INVOLVED

Not much, actually. In order to provide the basic communications features of Yenta, I only had to subclass two classes in the ACL – `CPPReadTask` and `CPPListenTask`. Implementing the auto-scan/confirm features required the construction of five more descendants of `CPPPeriodicTask`. The interface required about 40 other classes, but I'm not going into those in detail here. The approach I will take in going over all this is to talk about the two new classes and how they work, then show how all of the basic communications classes are used in the application. After that I will discuss the periodic task classes which implement the scan and confirm features. Figure 4 presents an overview of the tasks used by the Yenta application, which may help you to sort out who is doing what to whom.

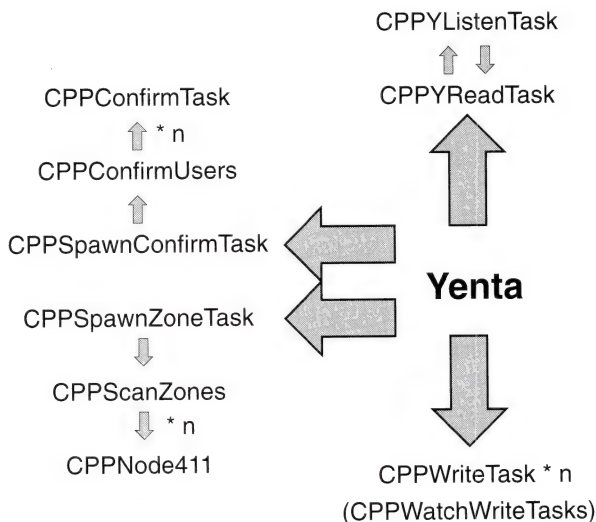


Figure 4. Yenta's Task Generation Map

CUSTOMIZING THE ACL

Back when I was discussing listening, reading, and writing, I noted that it could be beneficial to create links between the `CPPListenTask` and `CPPReadTask` classes so that when a listen completed it would spawn a read, and vice versa. That is essentially what the two new classes do. As a result, the only method we override in each class is `DoCompletedAction`. Both overloaded methods are listed below.

```
void CPPYListenTask::DoCompletedAction (void)
{
    CPPYReadTask *DoRead = NULL;

    CPPListenTask::DoCompletedAction();
    if (this->callResult == noErr)
    {
        DoRead = new CPPYReadTask (this->ourManager, 15, TRUE);
    }
}
```

```
DoRead->StartReadTask(this->sessionID, 100, NULL);
}

void CPPYReadTask::DoCompletedAction (void)
{
    PPCEndPBRec    EndRec;
    Boolean        AmITheOwner;
    CPPYListenTask *LTask = NULL;
    Handle TempHandle = this->GetData (FALSE, &AmITheOwner);

    CPPReadTask::DoCompletedAction();
    gTalkText->AppendItem(Hand2Ptr(TempHandle));

    EndRec.sessRefNum = this->sessionID; // close connection
    this->callResult = PPCEnd (&EndRec, FALSE);

    LTask = new CPPYListenTask (this->ourManager, 60, TRUE);
    LTask->StartListenTask(gOurPort, NULL);
}
```

The details of `CPPYListenTask`'s completion routine are fairly intuitive; if the listen task completes successfully it creates a new `CPPYReadTask` and starts it running on that session, using a period of 100 ticks.

The details of `CPPYReadTask`'s completion routine requires a bit more explaining, as it involves classes in other parts of the program. `gTalkText` is a queue which holds pointers to blocks of data – in this case messages received by the application. `DoCompletedAction` copies the received data into a pointer using `Hand2Ptr`, then adds it to the 'received message' queue. It then closes the connection and starts a new `CPPYListenTask` with a 60 tick period to wait for someone else to talk to us.

How It All Fits:

Listening and Reading

The nice thing about this tight link between the read and listen tasks is that once we have opened the communications port and posted the first listen task, everything else is done automatically; without our ever having to tell it to, the port is always engaged in reading or listening. The code which starts the whole thing off is part of the `s` method, and is shown below.

```
CPPMaBell        *gMaBell;
CPPTaskManager   *gSlaveDriver;

...
gMaBell = new CPPMaBell (TRUE);
if ((ErrCode = gMaBell->InitPPCToolbox()) != noErr)
{
    ErrorAlert (ErrCode, NULL);
    ExitToShell();
}

...
// create the lookup/read/write task manager
gSlaveDriver = new CPPTaskManager();

...
// open the port we will use to communicate through
if ((ErrCode = gMaBell->OpenCommunicationPort (ObjString,
                                                gAppName, &gOurPort)) != noErr)
{
    ErrorAlert (ErrCode, "\pCan't open a port to communicate with.");
    ExitToShell();
}

...
// Set up a ConnectionTask to handle communications
LTask = new CPPYListenTask(gSlaveDriver, 60, TRUE);
LTask->StartListenTask(gOurPort, NULL);
```

When the application is shutting down, we simply delete `gSlaveDriver` – which causes any outstanding tasks to be

TASTES LIKE CHICKEN!

Without secure software, even the most expensive hardware key is nothing but a chew toy.



You need to protect your software. Hardware vendors will try to convince you that hardware keys are the only secure method of protection. They're wrong.

Protection is only as effective as the software involved. Most hardware key vendors require that you, the developer, spend major resources protecting and obscuring the code which interacts with a key. Without this effort on your part, your software is vulnerable, even to inexperienced hackers.

At PACE, we understand that software is the most important component of a working protection system. For more than a decade we've developed low cost and secure software based protection schemes. Our MacEncrypt system will turnkey protect your application automatically, applying multiple layers of PACE proprietary encryption and self checking algorithms to your product.

MacEncrypt is a secure, flexible, compatible and low cost protection investment. No false promises, wasted development time, upset users or expensive chew toys.

Trust the protection of your software to the people who understand software. Call today to order your PACE Developers Kit.

PACE Anti-Piracy 1082 Glen Echo Ave., San Jose, CA 95125
Vox: (408) 297-7444 • Fax: (408) 297-7441 • AppleLink: PACE.AP
email: info@paceap.com • Web page: <http://paceap.com/pace.html>



P A C E
ANTI-PIRACY

"Our Japanese font supplier imposes strict copy-protection obligations on Adobe Systems. Because of this, we have been using PACE software protection on Adobe Type Manager, Japanese version and our other Japanese font products for almost 5 years. PACE's software solution provides us with effective security at a low cost. Our working relationship with PACE is excellent and their expert technical staff has always been helpful."

Paul Anderson
Senior Director, Pacific Rim
Adobe Systems, Inc.

aborted and deleted, closes gOurPort, and deletes gMaBell to shut down Appletalk.

Writing

Writing data to other users is also a fairly simple process. All writing is done from the 'Send' window, so I created a method within it called SendToUser, the details of which are shown below.

```
void CPPSendWindow::SendToUser(Ptr Text, Boolean ownsData,
                               CPPNodeInfo *SendTo)
{
    CPPWriteTask *TheTask;

    // have the write task open the connection
    TheTask = new CPPWriteTask (gSlaveDriver, 25, TRUE);
    TheTask->StartWriteTask (gOurPort, SendTo, Text, ownsData,
                            NULL, (OSType)'TEXT', (OSType)'YntA');
}
```

SendToUser simply creates a CPPWriteTask and uses the 'automatic connect' version of StartWriteTask to send the data to the specified user.

SendToUser is used by two other methods within CPPSendWindow - the method which responds to the user pressing the send button, and the method which sends the AutoReply string back to someone who sends us a message. Looking at these two methods will complete our coverage of

how writing is done within Yenta. Let's look at the AutoReply method first:

```
void CPPSendWindow::GotNewMessage (CPPNodeInfo *FromWhom)
{
    Ptr    TempPtr, OurIDStream;
    OSErr  ErrCode;

    // send autoreply message if feature is on
    if (gReplyString)
    {
        // Create a string with our name and address in it
        OurIDStream = gOurIdentity->InfoToStream();

        // create a pointer to hold our address & the reply string
        TempPtr = NewPtr (GetPtrSize(OurIDStream)
                        + gReplyString[0]);
        if ((ErrCode = MemError()) == noErr)
        {
            BlockMove (OurIDStream, TempPtr, GetPtrSize(OurIDStream));
            BlockMove (gReplyString+1,
                      TempPtr+GetPtrSize(OurIDStream),
                      *gReplyString);
            DisposPtr (OurIDStream);

            // send the autoreply to the user who sent us the message
            SendToUser (TempPtr, TRUE, FromWhom);
        }
    }

    // Add the user to the list
    if (this->UserList->AddNewUser(FromWhom))
        if (gPrefsInfo.playLogon)
            gLogonSound->PlaySound(TRUE);
}
```

THE FINAL PIECES

This method is called every time a message is taken from the message queue. The first part checks a `StringPtr` called `gReplyString`; if `AutoReply` is turned on, `gReplyString` points to the `AutoReply` message, otherwise it is `NULL`. Because the standard format for Yenta messages requires that the address of the sender be included with the message, we first use `CPPNodeInfo`'s `InfoToStream` method to convert our address to a pointer, copy the string and the address into another pointer, then ask `SendToUser` to deliver the message. Note that the write task is given the responsibility for deleting the pointer when it completes.

The method which responds to the user pressing the send button is named (rather predictably) `DoSendButton`. The first part of the method collects information about the text to be sent, then constructs the message, storing it in a variable called `TempPtr`. It also stores the total number of selected users in the variable `NumToSendTo`, then enters the following loop:

```
// send text to each highlighted user
while (this->UserList->NextSelectedCell(&whichUser))
{
    UserData = (CPPNodeInfo *)((*this->UserList)[whichUser]);
    if (UserData && !(UserData->Equals(gOurIdentity)))
        SendToUser (TempPtr, NumToSendTo == 1, UserData);
}
```

Each selected user's `CPPNodeInfo` object is extracted from the user list, and the data sent to each of them using the `SendToUser` method. Note that the write task is only given permission to dispose of the data if there is one user selected.

When you send to more than one user, the matter of disposing of the write data becomes a bit more complicated, because there are no hard-and-fast rules for determining which write task should actually be given permission to dispose of the data. You can't give permission to any one task in particular, since you have no guarantee that any particular task will complete after all of the others. Similarly, you can't let the application dispose of the memory directly after the loop, since not all the tasks may have completed by then. You could create a copy of the data for each user and give each task ownership of its copy, but consider the problem of sending 10K of data to 50 users; you tend to run out of memory rather quickly.

The solution which I came up with was to create a subclass of `CPPPeriodicTask` which could hold on to the data pointer, and wait for all of the write tasks to complete before disposing of it. This class, called `CPPWatchWriteTasks`, accomplishes this task by calling its `PTM`'s `HowManyTasksOfType` method with the name "CPPWriteTask" until the count reaches zero, then completing and deleting the pointer it was given. The following fragment comes directly after the 'send' loop shown above:

```
// keep track of 'write' data until all tasks complete
if (NumToSendTo != 1)
{
    WatchTask = new CPPWatchWriteTasks (gSlaveDriver, 60);
    WatchTask->StartWatchTask(TempPtr);
}
```

The last two features to discuss are the program's ability to automatically scan known zones for new users and to confirm the presence of known users. Each of these features required the creation of two subclasses of `CPPPeriodicTask`: one which does the actual work, and the other which simply triggers it every *n* minutes. Let's look at the details of the trigger classes first.

There are two trigger classes – `CPPSpawnZoneTask` and `CPPSpawnConfirmTask`. Each class has three elements in common: 1) a private variable of the type task it triggers (`CPPScanZones` or `CPPConfirmUsers` respectively), 2) a `Start???` method which initializes the private variable and enqueues the task, and 3) a `DoPeriodicTask` method which, when called, calls `Start???` on that private variable. Below is the code for one of their `DoPeriodicTask` methods; the other is identical in style.

```
void CPPSpawnZoneTask::DoPeriodicAction (void)
/* if the Scan task has completed, ask it to scan again; */

// call the inherited method to update frequency count
CPPPeriodicTask::DoPeriodicAction();
if (scanTask->hasCompleted)
    scanTask->StartScanZones (NULL);
```

A key feature of both trigger tasks is that they never complete. (What, never? No, never!) Both of them remain in the queue until they are either explicitly removed or until the application shuts down. Boring, but useful. Time to move on to the gruntwork classes – `CPPScanZones` and `CPPConfirmUsers`.

CPPSCANZONES

`CPPScanZones` is a fairly unassuming descendant of `CPPPeriodicTask`. It has a single unique method – `StartScanZones`, and three private variables, shown below:

```
class CPPScanZones : public CPPPeriodicTask {
public:
    ...
    void StartScanZones (CompletionProc DoProc);
private:
    CPPStringList *zoneList;
    CPPNode411 *lookupTask;
    long whichZone;
};
```

When `StartScanZones` is called, it copies the list of zones in the `Send` window into the `zoneList` variable, allocates the `lookupTask` object, and sets `whichZone` to 1. Its `DoPeriodicTask` method looks like this:

```
void CPPScanZones::DoPeriodicAction (void)
/* if the lookup has completed, advance to the next zone and start another lookup */
{
    StringPtr ZoneName;
    Str32 TypeStr;
    Str255 STemp;

    // call the inherited method to update frequency count
    CPPPeriodicTask::DoPeriodicAction();

    if (lookupTask->hasCompleted)
```



```

this->whichZone++;
if (this->whichZone <= this->zoneList->GetNumItems())
{
    ZoneName = (*zoneList)[this->whichZone];
    PStrCat (32, TypeStr, 2, "\p≈", gAppName);
    lookupTask->StartNodeLookup("\p=", TypeStr,
        ZoneName, 50, ProcessNodeLookupResults);
    PStrCat (255, STemp, 3, "\pScanning zone '",
        ZoneName, "\p' for new users.");
    SetStatusMessage (STemp, TRUE);
}
else
{
    this->hasCompleted = TRUE;
    this->callResult = noErr;
}
}
}

```

At each iteration, it gets the name of the next zone in the list and starts the CPPNode411 task looking for all objects in that zone which match the ACL naming convention, using the application name as the PortType (see the discussion of CPPMaBell's OpenCommunicationPort method). The global routine ProcessNodeLookupResults takes each node in the 'found' list and passes it to the user list, which then determines whether the node is already in the list or not.

CPPCONFIRMUSERS

CPPConfirmUsers is also a fairly unassuming task, with a declaration similar to that of CPPScanZones:

```

class CPPConfirmUsers : public CPPPeriodicTask {
public:
    ...
    void      StartConfirmUsers (CompletionProc DoProc);
private:
    CPPObjectList *nodeList;
    CPPConfirmTask *confirmTask;
    long          whichNode;
};

```

StartConfirmUsers copies the list of known addresses from the Send window into the nodeList variable, allocates the confirmTask object, and sets whichNode to 1. Its DoPeriodicTask method follows:

```

void CPPConfirmUsers::DoPeriodicAction (void)
{
    CPPNodeInfo *TheNode = NULL;
    Str255      STemp;
    StringPtr    UserName;

    CPPPeriodicTask::DoPeriodicAction();

    if (confirmTask->hasCompleted)
    {
        this->whichNode++;
        if (this->whichNode <= gUserList->GetNumItems())
        {
            TheNode = (CPPNodeInfo *)((*gUserList)[whichNode]);

            // tell the user we are confirming this connection
            UserName = ShortName (TheNode);
            PStrCat (255, STemp, 2, "\pSearching for ", UserName);
            SetStatusMessage (STemp, TRUE);
            NukePtr(UserName);

            confirmTask->StartNodeConfirm(TheNode,
                ConfirmCompletionProc);
        }
    }
}

```

```

else
{
    this->hasCompleted = TRUE;
    this->callResult = noErr;
}
}
}

```

In a manner similar to CPPScanZones, it iterates through its list, extracting each user address in turn and activating the confirm task. Note that it assigns a completion routine to CPPConfirmTask. This is done primarily so that we don't have to create a specific subclass of CPPConfirmTask. The completion routine simply asks the user list to delete the specified user if the NodeExists method returns FALSE (indicating that the confirm task failed to find the user on the network).

IN CONCLUSION

Believe it or not, we're done. As promised, I haven't discussed any of the interface classes/application framework which I used to put Yenta together. If you are comfortable with the TCL, you probably needn't bother, unless you want to work entirely in C++. If, like me, you find the TCL a bit baroque and unintuitive, you might want to look over the classes I've constructed. They don't provide as many fancy features as the TCL (embedded scrolling panes, etc.) but most of the classes map pretty directly onto the Macintosh user interface, which I think makes it easier to use. If there is enough interest, I may discuss parts of it at a later date. If anyone finds any bugs (bound to be in there, somewhere!), or comes up with any neat classes which add functionality to either the ACL or my interface class library, please snail/e-mail me and let me know. Feel free to modify, subclass, and experiment like mad. Good hacking to you all!

REFERENCES

Books

Inside AppleTalk, Second Edition. Addison-Wesley Publishing Company. Good overview of network topology and communication protocols.

Pierce, Michael. *Programming with Appletalk*. Addison-Wesley Publishing Company. Great overview of the nitty-gritty details; lots of 'how to' code.

Technical Manuals

Inside Macintosh, Vol II, chapter 10

Inside Macintosh, Vol V, chapter 28

Inside Macintosh, Vol VI, chapters 7 and 32

Inside Macintosh: Interapplication Communication, chapter 11

Software (available from Apple DTS)

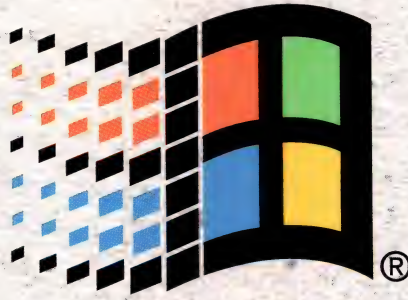
Network Watch (DMZ)

Neighborhood Watch

SC011.GetZoneList



Designed for



Microsoft®

Windows™ 95

hello

Designed for



Microsoft®
Windows™ 95

How ya doin'? They say opportunity breaking down the door. Windows™ 95 looks programming experience has prepared you to be on



The Scoop! You love the Mac®, Right?

Because early on, it let you develop cool new applications that you couldn't create anywhere else. Applications that delivered a compelling end-user experience. Applications that worked together more closely than ever before. We know; we feel the same way. That's why Microsoft was one of the first and strongest supporters of the Mac, even before it was launched. Last year we shipped more applications for the Mac than anyone else. We still support the Mac—and you should too.

Because you love the Mac experience

you should also look at Windows™ 95. Intuitive user interface, Plug-n-Play hardware, long file names...they're all there in Windows 95. Throw in pre-emptive multitasking, an integrated object model, and enterprise connectivity...and you've got a very powerful platform for creating a whole new generation of insanely great applications.

You're wondering how to get involved?

We're not talkin' about life in the slow lane here. When you turn the page you'll see a valuable offer from Dell that can help you join your friends in developing for Windows 95—today.

Get Back to the Mac®

WITH

Microsoft® Visual C++™ Cross-Development Edition for Macintosh®

You're asking yourself, OK, so I write all this new code for Windows™—how do I get that code back to the Mac? To make this possible, Microsoft recently released Microsoft® Visual C++™ Cross-Development Edition for Macintosh®, version 2.0. Visual C++ produces a native application for the Macintosh from code written to Windows™ 95's native Application Programming Interface, Win32®.

- ☛ **Recompile for the Mac:** Target both the Windows and Macintosh operating systems using just one set of source code written to Microsoft Foundation Classes or directly to the Microsoft Win32 API.
- ☛ **Call the System 7.x Toolbox directly:** Take advantage of unique Macintosh features such as Publish and Subscribe.
- ☛ **Familiar Macintosh interface:** Use the Visual C++ resource editor to create application resources that look like native applications on both Windows and the Macintosh.
- ☛ **Fast executables:** Generate 68K code with high performance using the Visual C++ optimizing C/C++ compiler.

Visual C++ delivers a complete solution for cross-platform development across Windows and the Macintosh. With it, you can use up to 90 percent of your code from Windows-based applications to create the same applications for the Macintosh, with the familiar Macintosh interface. It's the quickest way to bring out the best in both worlds.

To order, or for more information, call

800-719-5577

Designed for



Microsoft®
Windows 95

knocks. This time opportunity is to be a phenomenal success, and your Mac® of the best Windows™ 95 developers in the business.



When Manny, Jorge, and I founded Deneba in '84, we wanted to work with the coolest technology then, just as we do now. We built our company on the Mac, delivering high-quality graphics applications, like Canvas. With Windows 95, Microsoft has sweetened the pie with an OS architecture that's just too good to pass up. Built-in image-color matching, enhanced graphics, and 32-bit virtual memory management make it a great platform for Canvas. We're really gung-ho on Windows 95.

—Joaquin De Soto, Chief Executive Officer, Deneba Software



I co-founded Aldus in 1984 to develop PageMaker® on the Mac, but now at Shapeware I'm working mostly with Windows®. Knowing the Mac as well as I do, I have to admit I'm impressed by Windows 95: its memory management, 32-bit linear address space, and pre-emptive multi-tasking make development for Windows 95 a dream come true. With its new user interface, long file names, and Plug-n-Play hardware, you can deliver a great end-user experience to a very wide audience. Every Mac developer should consider supporting Windows 95, too.

—Dave Walter, Manager of Advanced Development, Shapeware



The transition to Windows 95 offers smaller, fast-moving companies the same great opportunities found in the early days of the Macintosh. Windows 95 is where the action is.

—Martin Mazner, former Publisher of MacUser Magazine



Professional experience in Windows 95 programming will significantly enhance the marketability of seasoned Macintosh developers.

—Dave Small, President, Scientific Placement

Help is just a phone call away:



[for once, some fine print you'll like]

The Microsoft® Developer Network (\$495)

An annual membership program that streamlines your access to Microsoft development information and technology.

- At least four quarterly updates to the Development Platform. You get it all: • Windows® 3.1 • Windows® for Workgroups 3.11 • Windows NT® 3.5 • Windows 95 • OLE • CMC • ODBC • TAPI • TCP/IP, • Finnish • Hungarian • Arabic • Thai, etc: it's all in there (the October '94 release of the Development Platform required over a dozen CD's) in an easy-to-use, well-organized form • Six bimonthly issues of the Developer Network News. • A 20% discount on Microsoft Press® books. • A \$20 one-time credit on CompuServe® connect charges.

Microsoft Visual C++ Subscription (\$499)

Your one source for the latest Visual C++ tools and information. It provides three updates per year that are delivered to you on CD-ROM, including: • Visual C++ version 2.0 development system for Windows and Windows NT • Visual C++ version 1.5 development system for Windows • Online documentation for all Visual C++ products • OLE Control Developer Kit • Visual C++ and MFC technical articles • Updates, including new features, sample code, ODBC drivers, enhancements to the Microsoft Foundation Class Library, recent technical articles, and bug fixes. The Visual C++ Subscription will keep you on the leading edge with the latest tools to develop for Windows 95 before it is released.

Microsoft Developer Relations Kit (Free)

To get information about our Developer Support services and products, order the Microsoft Developer Relations Kit. This kit has information about access to betas, the Developer Solutions Team, Testing Labs, the Multimedia Developer Relations Group, Usage of the Windows 95 logo, the CompuServe Information Service, Microsoft Support Network: Electronic Information Services, and much more:

To order your Microsoft Developer Relations Kit
or for more information, call:

800-719-5577

ask for reference part no. 098-57519m : 6:30am–5:30pm PST, Mon–Fri

Turn page for a valuable offer



Is this a great deal
or what?

YOU MAKE THE CALL!

The Dell® Developer System

COMPLETE WINDOWS™ 95 DEVELOPMENT SUITE



Dell OptiPlex™ 560/L, \$3,999



The Dell OptiPlex™ system will set you up with everything you need to start developing great applications for Windows™ 95. This system is fully configured as a Windows 95 development platform. With 60* Mhz Pentium™ power and speed, Plug-n-Play, advanced power management and much more.

As all developers know, without software, even the most powerful machine is just an expensive doorstop. So the Dell Developer System includes everything you need to get started with professional development for Windows.

Included is the Microsoft Developer Network, an annual membership program that delivers every single bit and byte of technical information plus the software developer kits and operating

systems you need. A \$495.00 value, you get one year's subscription FREE with the purchase of the Dell Development System.

Also included is the Microsoft® Visual C++™ Subscription which gives you the tools you need to get to work on Windows. You get Visual C++ 2.0 for Windows NT™ and Windows 95; Visual C++ 1.5 for Windows 3.1; the OLE Control Developers Kit...you name it, it's in there, all updated at least three times a year. A \$499.00 value, you get one year's subscription FREE with purchase of this system.

With these subscriptions you will get the Windows 95 SDK and Windows 95 specific tools, when they are released to developers.

You Get All This Great Stuff:

- » 24MB RAM
- » 1 MB Video Memory
- » 101 Key Performance Keyboard
- » Microsoft Mouse
- » Sound Blaster 16 with Dual Speed CD-ROM & Speakers
- » Dell Ultrascan 15" Trinitron
- » 528MB Hard Drive
- » 3.5" Floppy Drive
- » 3Com EtherLink III Combo
- » 3-year limited warranty*
- » Microsoft Windows NT Workstation V3.5
- » Microsoft MSDN Membership
- » Microsoft Visual C++ Subscription

DELL®

Dont miss out on this incredible value: Call and order the Dell Developer System today!

To order, call 800-627-9862 and ask for order # 300272

In Canada, call: 800-668-3021 Mon-Fri, 7am-9pm Central Time

*For even more powerful performance, a 90 Mhz Pentium system is available at extra cost. Contact Dell for more information.

Prices valid in the US only. Some products and promotions may not be available outside the US. Prices and specifications subject to change without notice. *For a complete description of Dell's 3-year limited warranty, please write to Dell USA LP, 2214 W Braker Lane Bldg 3, Austin TX 78758-4053 Attn: Warranty. Pentium is a trademark of Intel Corporation. Dell disclaims proprietary interest in the marks and names of others.



Using Low Priority Events in MacApp

Fixing a minor bug gets your priorities straight

Just like most standard Macintosh programs, MacApp has a main event loop, but as with many things, MacApp handles the gory details of the event loop for you while still giving you the flexibility to expand or improve upon it as needed. The focus of MacApp's event loop is MacApp's event list which usually contains commands but can also contain more generalized events. Commands and events posted to this list can have different priorities to change the order in which they are processed. The only problem is that MacApp 3.0 and 3.1 never actually process your low priority events.

This article gives a quick overview of the MacApp event list, explains why you might want to use an event with a low priority, and tells you how to fix MacApp-without modifying the MacApp source-so low priority events are properly processed.

INSIDE THE EVENT LIST

The MacApp event list is of type TEventList and is a data member, named

fEventList, of TApplication. TEventList contains objects of type TEvent and objects descended from TEvent including objects of type TCommand. Since TCommand is a descendent of TEvent, I will use the word "events" in this article to refer to both events and commands.

When you call PostAnEvent() or PostCommand(), the TEventHandler implementations of these two methods pass the event to the next event handler in the event handler chain until TApplication::PostAnEvent() gets the event and inserts it in fEventList sorted by priority. TApplication's main event loop method retrieves events from fEventList and handles the events by calling their Process() method. The highest priority events, those with their fPriority field set to kPriorityHighest, are retrieved before the lower priority ones. The priorities defined by MacApp are:

```
// Low priority commands are considered last
const short kPriorityLowest = 127;
const short kPriorityLow = kPriorityLowest - 32;
// Normal priority: command default priority
const short kPriorityNormal = 64;
const short kPriorityHigh = kPriorityNormal - 32;
//High priority commands take precedence
const short kPriorityHighest = 0;
```

If you wish you can use priority values which are between these constants. The default priority for events is kPriorityNormal.

Events of equal priority in fEventList are not necessarily processed on a First-In, First-Out basis. MacApp uses a binary search when inserting events in fEventList and inserts the event at the first event it finds of equal priority. If you post an event and there are already two or more events of equal priority in the list, their order in the list is indeterminate and hence their order of processing is indeterminate. This is not normally a problem since the typical MacApp application does not have that many

Harry Haddon – A MacApp user for over seven years, Harry works at Franklin & Marshall College writing applications encompassing topics from Homer to spectroscopy. Before becoming fascinated with things Macintosh, he worked as an electrical engineer designing microprocessor-based controls and marketed a C compiler he wrote for the Apple II. In his free time Harry enjoys folk music and plays the Anglo concertina, guitar, and banjo-ukulele. His email address is Harry@fandm.edu.

equal priority events in the list at one time, but it is something to consider if you're posting multiple commands to the list at the same time and the order of processing is important.

One command you'll always find in `fEventList` is the `TEventRetrieverCommand` that `MacApp` uses to fetch toolbox events from the toolbox's Event Manager. The initialization method `IApplication()` creates this command with a priority of `kPriorityLow` and posts it to `fEventList`. The command stays in the list as long as the application is running, and its sole job is to check for toolbox events. Since `TEventRetrieverCommand` has a lower priority than normal, `MacApp` does not process it until after it processes the events in the list that have a normal priority. Thus `MacApp` won't fetch any more events from the toolbox queue until after it has processed all of the normal priority events and commands in `fEventList`.

`TEventRetrieverCommand::DoIt()` checks for toolbox events by calling `gApplication->PollToolboxEvent()` which calls the toolbox trap `WaitNextEvent()`. If a toolbox event is available, it is encapsulated in a `TToolboxEvent` and processed by `MacApp`. If no toolbox event is available and `TApplication.AllowApplicationToSleep` is true, `MacApp` figures out the various sleep parameters such as the sleep time and calls `WaitNextEvent()` to wait for the next toolbox event.

This all works great unless you try to post an event with a priority of `kPriorityLow` or lower. Then you will find that the `TEventRetrieverCommand` in `fEventList` acts as a road block for low priority events. Because it was posted first, it is processed before all events of the same priority (`kPriorityLow`). If no toolbox events are available from the Macintosh event queue, the `TEventRetrieverCommand` puts the application to sleep, preventing the processing of any low priority events remaining in `fEventList`. If a toolbox event is available, `MacApp` processes it, as it should, leaving no opportunity for the processing of low priority events.

WHY USE LOW PRIORITY EVENTS?

I ran into the bug with low priority events when I was developing a client application that fetches data from a server application. I used a descendent of `TClientCommand`, `MacApp`'s class for sending an Apple event and processing its reply, to fetch the data from the server. The server collects new data at the rate of 10 samples per second and the client needs to be updated at least several times a second so as soon as a reply is received, the client posts another `TClientCommand` to fetch the next chunk of data.

My `TClientCommand` needed to be a lower priority than toolbox events so that the view that was changed by the `TClientCommand` would be updated via an update event before the next `TClientCommand` was processed. I also wanted the application to process toolbox events before it did the `TClientCommand` so that the application would be responsive to user actions such as mouse clicks. Experimentation with the `TClientCommand`'s priority set to `kPriorityNormal` on a slower Macintosh confirmed that being able to set its priority lower was a worthy goal.

You may have a similar situation where a low priority command would fit the bill. Remember that low priority commands really aren't background or idle commands: they do not execute until after higher priority events have executed, but once they begin execution they can hog CPU cycles as much as any other event. If they take too much time to execute they can slow down the processing of user actions and create a less than enjoyable experience for your user. Design your commands accordingly.

FIXING THE LOW PRIORITY EVENT BUG

I came up with a fairly simple fix that I have used with `MacApp` 3.0.1. This fix will probably also work with 3.1, since it appears that the relevant sections of code have not changed from 3.0 to 3.1. It is not a perfect fix in that events with the very lowest priority, `kPriorityLowest`, are still not processed, but this is not really a problem since you can use a priority of `kPriorityLowest-1` for your lowest priority, and it will work fine.

The original `TEventRetrieverCommand`, which is installed by `IApplication`, is left in `fEventList` but its priority is changed to `kPriorityLowest`. This still allows the application to sleep-a Good Thing in the Macintosh world of cooperative multi-tasking-but it does not go to sleep until after all other commands are given a chance to execute. I changed the priority of `TEventRetrieverCommand` in `IMyApplication()` after calling `IApplication()`:

```
TEventRetrieverCommand *originalEventRetriever;
originalEventRetriever =
    (TEventRetrieverCommand *) fEventList->At(1);
originalEventRetriever->fPriority = kPriorityLowest;

if (qDebug && !originalEventRetriever->IsMemberClass(
    GetClassIDFromName("TEventRetrieverCommand")))
    ProgramBreak("First command in fEventList is not \
a TEventRetrieverCommand!");
```

This code doesn't look for the `TEventRetrieverCommand` on the event list but just assumes that it's the first command on the list. The debug check will warn me if this isn't true in future versions of `MacApp`. (Hopefully Apple will fix this in `MacApp` 3.5 and we won't need this fix at all anymore.)

To keep processing toolbox events at `kPriorityLow`, I declared a new command that is a descendant of `TEventRetrieverCommand`. This command checks for toolbox events but never sleeps. It is posted at `kPriorityLow` to replace the original `TEventRetrieverCommand` that was demoted to `kPriorityLowest`.

```
class TNoSleepEventRetrieverCommand :
    public TEventRetrieverCommand {
public:
    TNoSleepEventRetrieverCommand();
    // Empty constructor to satisfy compiler.

    virtual pascal void TNoSleepEventRetrieverCommand(
        CommandNumber itsCommandNumber);
    // Initialize the EventCommand procedurally.
    virtual pascal Boolean IsReadyToExecute();
    // override
    // Return true when event available
```


NEW

absoft

development tools and languages

NEW

C++ for Power Macintosh

- Native Power Macintosh C++ compiler
- ANSI and K&R C
- 100% Plum Hall Validated
- Templates
- High Speed Math Library

- Fx™ multi-language debugger
- Power-Link — Absoft's native linker
- Apple's MPW development environment
- Link compatible with Absoft's F77 SDK
- Open, flexible development environment

Available Now: \$399.00 Complete

Tel: (810) 853-0050 • Fax: (810) 853-0108 • AppleLink: absoft • Internet: c@absoft.com

```
virtual pascal void DoIt();  
// Retrieve and process an event without sleeping  
};
```

I put the declaration for TNoSleepEventRetrieverCommand in the header file that contains the declaration for TMyApplication.

I put the definitions for its methods in the .cp file that contains the methods of TMyApplication. The initialization method INoSleepEventRetrieverCommand() just calls IEventRetrieverCommand() and then sets the command's priority:

```
#pragma segment ASelCommand  
pascal void  
TNoSleepEventRetrieverCommand::INoSleepEventRetrieverCommand(  
    CommandNumber itsCommandNumber)  
{  
    this->IEventRetrieverCommand(itsCommandNumber);  
  
    // Let more important stuff happen first  
    fPriority = kPriorityLow;  
}
```

Its IsReadyToExecute method returns true whenever a toolbox event is available:

```
#pragma segment ARes  
pascal Boolean  
TNoSleepEventRetrieverCommand::IsReadyToExecute()  
{  
    EventRecord theEvent;
```

```
    return EventAvail(gApplication->fMainEventMask, theEvent);  
}
```

When IsReadyToExecute() returns true, MacApp calls the command's DoIt() method. The DoIt() for TNoSleepEventRetrieverCommand is just like DoIt() for TEventRetrieverCommand except it calls PollToolboxEvent() with the parameter allowApplicationToSleep set to false so the application doesn't go to sleep on us:

```
#pragma segment ASelCommand  
pascal void TNoSleepEventRetrieverCommand::DoIt()  
{  
    gApplication->PollToolboxEvent(FALSE);  
    // FALSE = never sleep  
}
```

The TNoSleepEventRetrieverCommand is created and posted in TMyApplication after the priority of the original TEventRetrieverCommand is changed:

```
TNoSleepEventRetrieverCommand *aEventCommand =  
    new TNoSleepEventRetrieverCommand;  
aEventCommand->INoSleepEventRetrieverCommand(cNoCommand);  
this->PostAnEvent(aEventCommand);
```

That's it. With these fixes in place you can post a command with a priority of kPriorityLow or lower, and MacApp will process it as it should.





A Quick Trip Into the Depths

ResError Considered Harmful?

Nosy is over 10 years old, having been introduced to the world in Nov '84 and first shipped in Jan '85. It has been a few years since I have written an article for MacTech. Since '84, the Mac has become more powerful than the mainframes I used to work on, and the system has advanced in utility and complexity. Nosy hasn't changed too much, and some pseudo alternatives to it, such as the ResEdit CODE Editor have come along, but I used it recently to get the global view of a piece of the ROM.

The particular example I'll show here came to me as a bug in my debugger that showed up on a IIfx running System 7.5. After some investigation, it turned out that the problem was some less-than-robust code in Sound Manager 3.0. The crux of the problem was that, despite what Inside Macintosh or today's equivalent of it says, the Resource Manager doesn't always return a non-zero value of ResErr when it doesn't find a requested resource. More on this later.

The original problem presented to me was that a user was trying to debug an INIT and my debugger was barfing. To find the bug, I set up my normal debugger bug-finding environment, in which I run two copies of my Debugger.

The first one debugs the second one, and if both fail, I have Macsbug running "behind" the first one. For the curious, the magic behind having one

debugger debug a second copy of itself is to do a complete context switch of low memory, which includes the exception vectors (0-\$FF) and the Macintosh globals (\$100-\$1E00) when entering The Debugger.

Using this method, I found that someone was trying to do a `_CloseResFile` on me. It took a few hours to trace it back into the Sound Manager.

At this point let me mention a trick I have used over the years to get a better grip on where one is in a random blowup. One of the most important pieces of information is the stack crawl. It answers the question of how and sometimes why the program is where it is. Many times the stack crawl is less than informative, and those who use Macsbug know well that the ROM part of it can be useless. Given this, how can we recover any information about how the program got to the point it is currently at? The basic answer is to try and repair the current

**“ Search your code for ResError calls
and convince yourself
that they serve some purpose...”**

address/bus error by fixing up a register or so (e.g. by putting \$0 or \$40801000 into the register which has a bus error value that the current instruction is attempting to read from or write to), advancing the Program Counter, and then carefully single stepping our way out of the current procedure and 'up the stack' so we can get an idea of where we came from.

In this case, this method worked for me, and I found myself in Sound Manager code, so the next problem was to get a decent disassembly of it so I could figure out why it was trying to do a `CloseResFile`.

After some disassembly work in Nosy, I found myself staring at the following two procedures, one which I have named `open_Snd_Prefs`:

		<code>open_Snd_Prefs</code>
<code>B1B10: 4E56 FFB4</code>	<code>'NV..'</code>	<code>LINK A6,#-\$4C</code>
<code>B1B14: 48E7 0308</code>	<code>'H...'</code>	<code>MOVEM.L D6-D7/A4,-(A7)</code>
<code>B1B18: 598F</code>	<code>'Y.'</code>	<code>SUBQ.L #4,A7</code>
<code>B1B1A: 2F3C 5354 5220</code>	<code>'/<STR '</code>	<code>PUSH.L #'STR '</code>



Call
1-800-282-2732
to order today

You Can't Buy This CD in any Store

More than 30 popular SDKs—one low price

What's the easiest and least costly way to make the most of the new features enabled by Macintosh system software extensions?

In a word (or five), the Mac™ OS Software Developer's Kit (SDK).

The Mac OS SDK helps you support the entire Mac OS family, by offering instant access to *more than 30* of Apple's most proven and popular SDKs. And whereas these SDKs previously sold separately for nearly \$2,000, now you can have them all on one CD—for a one-year subscription price of only \$299.



Mac OS
Software Developer's Kit

You won't find this special Mac OS SDK in any store or catalog. It's available *only* from APDA, Apple's source for development tools. Nor will you be left on your own as soon as you make the purchase. Every quarter you'll receive an automatic update—ensuring you always have the most current information as well as an understanding of how to add support for advanced system extensions in your software.

A typical SDK provides the system software extension; programming interfaces and libraries; sample code; and technical documentation. Here's a listing of the SDKs that are included in your first CD mailing:



Apple Guide
Apple Open Collaboration
Environment (AOCE)
Apple Remote Access
Apple Remote Access Modem
Apple Shared Library Manager
AppleScript
AppleSearch
AppleShare API

AppleTalk Wide Area
ColorSync
Communications Toolbox
Control Strip
Designing PCI Cards &
Drivers
File System Manager
Installer
Macintosh Drag and Drop

Macintosh Easy Open
MacODBC
MacOSI Connection
MacSNMP
MacTCP
MacX.400
MacX25
MIDI Management Tools
Network Software Installer

Open Transport
PlainTalk
QuickDraw GX
QuickTime
Sound Manager
Telephone Manager
Thread Manager
XTND



How to Order Mac OS SDK

To order by phone: call **1-800-282-2732** (U.S.)
1-800-637-0029 (Canada), or **(716) 871-6555** (International)
Monday through Friday, 7am to 5pm PST.
R0603LL/A \$299 per annual subscription (four mailings).

To order by mail: send a check or money order for \$299 payable to APDA/Apple Computer, Inc., plus sales tax and \$20 for shipping and handling (for U.S. orders) to: APDA, Apple Computer, Inc., P.O. Box 319, Buffalo, NY 14207-0319. For international orders, please call APDA for shipping and handling charges. Allow 5-10 days for delivery within the U.S.


```

B1B20: 3F3C BF48      '?<.'H'      PUSH    #$BF48
B1B24: A9A0            '...'      _GetResource
; (theType:ResType; ID:INTEGER):Handle

B1B26: 285F          '(_.'      POP.L    A4
B1B28: 200C          '...'      MOVE.L   A4,D0
B1B2A: 660A          10B1B36 BNE.S    mnn_1
B1B2C: 558F          'U.'      SUBQ.L   #2,A7
B1B2E: A9AF          '...'      _ResError ;:OSErr
B1B30: 3E1F          '>.'      POP      D7
; ROM returns ResErr = 0 !! *****
B1B32: 6000 0094      10B1BC8 BRA      mnn_4
no error from RsrcMgr, so we exit with D0 = 0, but we haven't opened the Prefs file!!!

B1B36: 558F          'U.'      mnn_1 SUBQ.L   #2,A7
B1B38: 3F3C 8000      '?<..'      PUSH    #$8000
B1B3C: 2F3C 7072 6566 '/<pref'    PUSH.L   #'pref'
B1B42: 7001          'p.'      MOVEQ    #1,D0
B1B44: 1F00          '...'      PUSH.B   D0
B1B46: 486E FFFE      200FFFE PEA      wnn_4(A6)
B1B4A: 486E FFFA      200FFFA PEA      wnn_3(A6)
B1B4E: 7000          'p.'      MOVEQ    #0,D0
B1B50: A823          '...'      _AliasMgr
; (D0/selector:INTEGER)
B1B52: 3E1F          '>.'      POP      D7
B1B54: 6672          10B1BC8 BNE.S    mnn_4
B1B56: 204C          'L.'      MOVEA.L  A4,A0
B1B58: A029          '.)'      _HLock   ;(A0/h:Handle)
B1B5A: 558F          'U.'      SUBQ.L   #2,A7
B1B5C: 3F2E FFFE      200FFFE PUSH     wnn_4(A6)
B1B60: 2F2E FFFA      200FFFA PUSH.L   wnn_3(A6)
B1B64: 2F14          '...'      PUSH.L   (A4)
B1B66: 486E FFB4      200FFB4 PEA      wnn_2(A6)
B1B6A: 303C 0001      'O<..'      MOVE     #1,D0
B1B6E: AA52          'R.'      _HighLvlFSDisptch
; (D0/selector:INTEGER)
B1B70: 3E1F          '>.'      POP      D7
B1B72: 0C47 FFD5      'G..'      CMPI     #-43,D7
B1B76: 6622          10B1B9A BNE.S    mnn_2
B1B78: 4A2E 000B      200000B TST.B    param2(A6)
B1B7C: 671C          10B1B9A BEQ.S    mnn_2
B1B7E: 486E FFB4      200FFB4 PEA      wnn_2(A6)
B1B82: 2F3C 7361 6420 'O<sad '    PUSH.L   #'sad '
B1B88: 2F3C 7072 6566 '/<pref'    PUSH.L   #'pref'
B1B8E: 70FF          'p.'      MOVEQ    #-1,D0
B1B90: 3F00          '?.'      PUSH     D0
B1B92: 303C 000E      'O<..'      MOVE     #14,D0
B1B96: AA52          'R.'      _HighLvlFSDisptch
B1B98: 4247          'BG'      CLR      D7
B1B9A: 4A47          'JG'      mnn_2 TST      D7
B1B9C: 662A          10B1BC8 BNE.S    mnn_4
B1B9E: 558F          'U.'      SUBQ.L   #2,A7
B1BA0: 486E FFB4      200FFB4 PEA      wnn_2(A6)
B1BA4: 7003          'p.'      MOVEQ    #3,D0
B1BA6: 1F00          '...'      PUSH.B   D0
B1BA8: 303C 000D      'O<..'      MOVE     #13,D0
B1BAC: AA52          'R.'      _HighLvlFSDisptch
B1BAE: 3C1F          '<.'      POP      D6
B1BB0: 0C46 FFFF      'F..'      CMPI     #-1,D6
B1BB4: 6608          10B1BBE BNE.S    mnn_3
B1BB6: 558F          'U.'      SUBQ.L   #2,A7
B1BB8: A9AF          '...'      _ResError ;:OSErr
B1BBA: 3E1F          '>.'      POP      D7
B1BBC: 600A          10B1BC8 BRA.S    mnn_4
B1BBE: 206E 000C      200000C MOVEA.L  param1(A6),A0
B1BC2: 3086          'O.'      MOVE     D6,(A0)
B1BC4: 7000          'p.'      MOVEQ    #0,D0
B1BC6: 6002          10B1BCA BRA.S    mnn_5
B1BC8: 3007          'O.'      mnn_4 MOVE     D7,D0
B1BCA: 4CEE 10C0 FFA8 200FFA8 mnn_5 MOVEM.L  wnn_1(A6),D6-D7/A4
B1BD0: 4E5E          'N^'      UNLK     A6
B1BD2: 4E75          'Nu'      RTS

B1CA6: 4E56 FFFE      'NV..'      proc4742 LINK    A6,#-2
B1CAA: 48E7 0738      'H..'      MOVEM.L  D5-D7/A2-A4,-(A7)
B1CAE: 246E 000C      200000C MOVEA.L  param2(A6),A2
B1CB2: 266E 0008      2000008 MOVEA.L  param3(A6),A3
B1CB6: 7000          'p.'      MOVEQ    #0,D0

B1CB8: 1012          '...'      MOVE.B   (A2),D0
B1CBA: 4A80          'J.'      TST.L    D0
B1CBC: 6604          10B1CC2 BNE.S    mnq_1
B1CC0: 7ECE          '...'      MOVEQ    #-50,D7
B1CC2: 6074          10B1D36 BRA.S    mnq_6
B1CC4: 200B          '...'      mnq_1 MOVE.L   A3,D0
B1CC6: 6604          10B1CCA BNE.S    mnq_2
B1CC8: 7ECE          '...'      MOVEQ    #-50,D7
B1CCA: 606C          10B1D36 BRA.S    mnq_6

B1CCE: 558F          'U.'      mnq_2 SUBQ.L   #2,A7
B1CCC: A994          '...'      _CurResFile ;:RefNum
B1CC6: 3C1F          '...'      POP      D6
B1CD0: 486E FFFE      200FFFE PEA      wnn_2(A6)
B1CD4: 7000          'p.'      MOVEQ    #0,D0
B1CD6: 2F00          '...'      PUSH.L   D0
B1CD8: 4EBA FE36      10B1B10 JSR      open_Snd_Prefs
B1CDC: 3E00          '>.'      MOVE     D0,D7
B1CDE: 504F          'PO'      ADDQ     #8,A7
B1CE0: 670A          10B1CEC BEQ.S    mnq_3
; if no error then get 'sysb' resource
B1CE2: 3F06          '?. '      PUSH     D6
B1CE4: A998          '...'      _UseResFile
; (refNum:RefNum)
B1CE6: 3D47 0014      2000014 MOVE     D7,funRs1t(A6)
B1CEA: 604E          10B1D3A BRA.S    mnq_7

B1CEC: 598F          'Y.'      mnq_3 SUBQ.L   #4,A7
B1CEE: 2F2E 0010      2000010 PUSH.L   param1(A6)
B1CF2: 2F0A          '...'      PUSH.L   A2
B1CF4: A820          '...'      _Get1NamedResource
; (theType:ResType; name:Str255):Handle
B1CF6: 285F          '(_.'      POP.L    A4
B1CF8: 558F          'U.'      SUBQ.L   #2,A7
B1CFA: A9AF          '...'      _ResError ;:OSErr
B1CFC: 3E1F          '>.'      POP      D7
B1CFE: 200C          '...'      MOVE.L   A4,D0
B1D00: 660A          10B1D0C BNE.S    mnq_4
; oops, didn't get it, close the res file
B1D02: 4A47          'JG'      TST      D7
B1D04: 6626          10B1D2C BNE.S    mnq_5
B1D06: 3E3C FF40      'X<.@'      MOVE     #$FF40,D7
B1D0A: 6020          10B1D2C BRA.S    mnq_5

B1D0C: 204C          'L'      mnq_4 MOVEA.L  A4,A0
B1D0E: A025          '...'      _GetHandleSize
; (A0/h:Handle);D0/Size
B1D10: 2A00          '...'      MOVE.L   D0,D5
B1D12: 3E38 0220      '$220'      MOVE     MemErr,D7
B1D16: 6614          10B1D2C BNE.S    mnq_5
B1D18: 204B          'K'      MOVEA.L  A3,A0
B1D1A: 2005          '...'      MOVE.L   D5,D0
B1D1C: A024          'S'      _SetHandleSize
; (A0/h:Handle; D0/newSize:Size)
B1D1E: 3E38 0220      '$220'      MOVE     MemErr,D7
B1D22: 6608          10B1D2C BNE.S    mnq_5
B1D24: 2054          'T'      MOVEA.L  (A4),A0
B1D26: 2253          'S'      MOVEA.L  (A3),A1
B1D28: 2005          '...'      MOVE.L   D5,D0
B1D2A: A22E          '...'      _BlockMove
; (A0/srcPtr,A1/destPtr:Ptr;
B1D2C: 3F2E FFFE      200FFFE mnq_5 PUSH     wnn_2(A6)
B1D30: A99A          '...'      _CloseResFile
; (refNum:RefNum)
B1D32: 3F06          '?. '      PUSH     D6
B1D34: A998          '...'      _UseResFile
; (refNum:RefNum)

B1D36: 3D47 0014      2000014 mnq_6 MOVE     D7,funRs1t(A6)
B1D3A: 4CEE 1CE0 FFE6 200FFE6 mnq_7 MOVEM.L  q_1(A6),D5-D7/A2-A4
B1D40: 4E5E          'N^'      UNLK     A6
B1D42: 205F          '...'      POP.L    A0
B1D44: 4FEF 000C      'O...'      LEA      12(A7),A7
B1D48: 4ED0          'N.'      JMP      (A0)

```

Referring to the procedure I named open_Snd_Prefs, it sets up a Link frame and calls _GetResource to get the value of a 'STR' resource with id = BF48. I rummaged around the System

file in Resorcerer and found that this string had the value of "Sound Prefs". The code then checks to see that it gets the string, and the rest of the procedure uses the Alias Mgr, etc to attempt to find its prefs file somewhere in the system folder. Apparently this procedure exits with 0 if it has found and opened the prefs file or a non-zero return when it has not. The caller futzes around to get the value of some magic resource from it and then closes the prefs file via _CloseResFile. So how, you may ask, could anything go wrong with this simple code?

Well, my debugger disconnects itself from the System file. That is, inside my Debugger, the System file is *not* on the chain of resource files to be searched, and, in almost all cases, it contains copies of the necessary resources that would normally come from the System file. The reason for this is that an open resource file contains a field for each resource that points to the copy of it in memory when the resource is opened. In order to avoid having to switch the values of these fields when transitioning between The Debugger and user, I chose the time efficient method of duplicating the necessary resources inside my debugger.

With this fact in hand, lets look at open_Snd_Prefs again. In C the source code would look like:

```
OSErr open_Snd_Prefs() {
    handle h = GetResource('STR ', $BF48); // get name of Sound Prefs file
    if( ! h )
        return( ResError ); // assumes the Resource Mgr returns a non-zero error
    {
        // locate and Open sound prefs file, ...

        return( 0 )
    }
}
```

As I mentioned at the beginning, the resource manager may return a NIL handle, but it rarely returns a non-zero value of ResError.

The upshot of all this is that the caller got a 0 error return and thought that the sound prefs file was open. When it did the Get1NamedResource, and it failed, it then tried to close the current resource file, which in this case was my Debugger, and things went to hell from there.

In retrospect, a more robust way to specify and code the routine might be to have it return the (FCB) refNum of the sound prefs file that it opened or 0. Then the logic in the calling proc would be somewhat cleaner and less subject to failure.

Rewriting, the revised code would be:

```
int open_Snd_Prefs() {
    int refNum = 0;
    handle h = GetResource('STR ', $BF48); // get name of Sound Prefs file
    if( ! h )
    {
        // use Alias Mgr, etc to locate and open Sound Prefs resource file
        refNum = ?? // some value returned by High.vlFSDisptch
    }
    return( refNum ) }
```

As a last thought, I suspect that the buggy version of the code has been distributed by DTS as sample code for you to use, and in my humble opinion, it leaves something to be desired. Not because it is intrinsically wrong, but because the Macintosh Resource Manager is inconsistent about returning a

Attention Scripters! ScriptWizard ...the essential scripting tool

ScriptWizard™ brings professional script editing, testing and debugging facilities to AppleScript™. Improve your scripting productivity with this powerful and intuitive tool!

4 1/2 mice - MacUser UK
4 stars - MacWorld UK

• Watch Variables

Variable-watcher shows a complete list of variables and script properties.

• Find & Replace

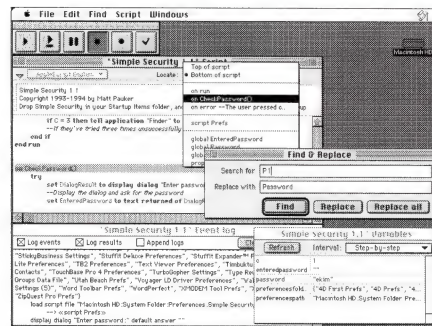
An irreplaceable tool for complex script development and modification.

• Step-Execution

Step-execution of scripts allows you to step through scripts one Apple event at a time.

• Easy Script Navigation

The "locate" pop-up shows all properties and handlers defined in your script, and instantly jumps to the one you choose.



"Its debugging capabilities are a 'must have' for any serious scripter"

- Dan Shafer: Author, Consultant and Scripting Advocate

only

\$99



US Offices:

P.O. Box 700237 San Jose, CA 95170-0237
Phone: (408)253-7199 Fax: (408)252-2378

European Offices:

P.O. Box 116 ST ALBANS, Herts, AL12RL, UK
Phone: +44 727 844232 Fax: +44 727 856139

non-zero value of ResError.

SOME THINGS THAT YOU CAN TRY AT HOME:

- 1) Search your code for ResError calls and convince yourself that they serve some purpose, or that you won't do anything really stupid if it returns 0 when it should not.
- 2) If you are using a class library such as MacApp, Metrowerks PowerPlant or TCL, then search it for ResError calls.
- 3) Selectively remove resources from your product and see what kind of stupid things it does.
- 4) Selectively remove resources from some one else's product and see what kind of stupid things it does. Submit bug reports.

An alternative title for this article could have been "ResError Considered Harmful". During the language wars of the 60's and 70's, I believe that Dijkstra threw the first stone by writing an article in SIGPlan, the ACM's Special Interest Group of Programming Languages titled "Goto Considered Harmful". His thesis was that the use of the goto statement made for poor programming style, etc. Over the next few years, the wars escalated to the point where I suspect that someone wrote an article titled "Harmful Considered Harmful".

[We welcome your comments, feedback, and debugging tales of woe and intrigue at editorial@xplain.com - Ed stb]





By Mike Scanlin, Mountain View, CA

SYMBOLIZE

This month's challenge was anonymously suggested. The goal is to make the output of a non-symbolic disassembler into a symbolic disassembly. The output file of the non-symbolic disassembler looks like this:

```
0006CDDE RTS
0006CDE0 LINK A6, #0000
0006CDE4 MOVE.L A4, -(A7)
0006CDE6 JSR 0006CE12
etc.
```

The symbol file you are given looks like this (and, yes, the real file I use for testing will have the LowMem and HiMem values in it so that you'll be able to find a symbol for any address):

```
00000000 LowMem
0006CDE0 Foo
0006CDE0 MyFunction
0006CE12 MyOtherFunction
FFFFFFF HiMem
```

Your job is to take every 8 byte hex value in the input disassembly and look it up in the symbol file and then substitute the symbol (and offset) for the value. If you ran your routine on the above fragment then the output would be this:

```
[Foo ] RTS
[MyFunction ] LINK A6, #0000
[MyFunction+4 ] MOVE.L A4, -(A7)
[MyFunction+6 ] JSR MyOtherFunction
```

The prototype of the function you write is:

```
void
Symbolize(inputFile, symbolFile, outputFile, symLength)
FILE *inputFile;
FILE *symbolFile;
FILE *outputFile;
unsigned short symLength;
```

InputFile is a standard C input stream containing the non-symbolic disassembly (as ASCII text). SymbolFile is a standard C input stream containing addresses (as ASCII text, not binary) and symbols (sorted by address, lowest to highest). The first symbol is for address 00000000 and the last symbol is for address FFFFFFFF, as shown in the example above. OutputFile is a standard C output stream that you send the symbolized disassembly to (also ASCII text). There is no need to fopen or fclose any of these streams (my test bench program will do it for you).

SymLength is the number of characters between the '[' and ']' in the outputFile (which will range from 12 to 32). All expressions of the form <symbolName>+<offset from symbol in base 10> should be exactly symLength characters long. Pad with spaces on the right if it's shorter and remove characters from the right side of symbolName if it's longer (always have the complete offset, unless it's zero). SymLength is 13 in the example above.

The largest symbolFile you'll receive is 512K and the largest inputFile you'll receive is 50K. You can assume that you'll have enough space for the outputFile. Before returning, your routine should dispose of any memory it might allocate.

Unlike some previous Challenges where you were allowed to write an untimed Initialize routine, there is no Initialize routine this month. The time it takes you to parse the symbolFile will be included in your overall time. The average inputFile will be 20K and contain 800 addresses to look up. The average symbolTable will contain 2000 symbols.

E-mail me if you have any questions. Have fun.

TWO MONTHS AGO WINNER

Congratulations to Challenge Champion **Bob Boonstra** (Westford, MA) for earning his sixth win in the Rubik's Cube

THE RULES

Here's how it works: Each month we present a different programming challenge here. First, you write some code that solves the challenge. Second, optimize your code (a lot). Then, submit your solution to MacTech Magazine. We choose a winner based on code correctness, speed, size and elegance (in that order of importance) as well as the postmark of the answer. In the event of multiple equally-desirable solutions, we'll choose one winner at random (with honorable mention, but no prize, given to the runners up). The prize for each month's best solution is \$50 and a limited-edition "The Winner! MacTech Magazine Programming Challenge" T-shirt (not available in stores).

To help us make fair comparisons, all solutions must be in ANSI compatible C (e.g. don't use Think's Object extensions). Use only pure C code. We disqualify any entries with any assembly in them (except for challenges specifically stated to be in assembly). You may call any routine in the Macintosh toolbox (e.g., it doesn't matter if you use NewPtr instead of malloc). We test entries with the FPU and 68020 flags turned off in THINK C.

We time routines with the latest THINK C (with "ANSI Settings", "Honor 'register' first", and "Use Global Optimizer" turned on), so beware if you optimize for a different C compiler. **Limit your code to 60 characters wide.** This helps with e-mail gateways and page layout.

We publish the solution and winners for this month's Programmers' Challenge two months later. All submissions must be **received by** the 10th day of the month printed on the front of this issue.

Mark solutions "Attn: Programmers' Challenge Solution" and send them via e-mail - Internet progchallenge@xplain.com, AppleLink MT.PROGCHAL, CompuServe 71552,174 and America Online MT PRGCHAL. Include the solution, all related files, and your contact info. If you send via snail mail, send a disk with those items on it; see "How to Contact Us" on p. 2.

MacTech Magazine reserves the right to publish any solution entered in the Programming Challenge of the Month. Authors grant MacTech Magazine the non-exclusive right to publish entries without limitation upon submission of each entry. Authors retain copyrights for the code.

Challenge. This is a special month for Bob because he has decided to retire from Programming Challenges and become the first person to enter the Programmer's Challenge Hall of Fame. Bob included this note along with his entry:

Should I be so fortunate as to win, I would like to announce my retirement from regular Programmer's Challenge competition. I may enter occasionally, particularly for any assembly language challenges or PowerMac challenges, but I want to devote more time to other pursuits. Besides, it's time to give someone else a chance. I have truly enjoyed the Challenges, and I commend Mike for keeping people focused on efficiency.

-- Bob Boonstra

Well, Bob, thanks for playing. We've all enjoyed learning optimization tricks from a master programmer during the last couple of years. I'm sure it will be a while before anyone collects as many wins as you have to take away your title.

Here are the times and code sizes for each entry. Numbers in parens after a person's name indicate how many times that person has finished in the top 5 places of all previous Programmer Challenges, not including this one:

Name	time	code+data
Bob Boonstra (13)	79	72,892
Ernst Munter (4)	156	4,526
Allen Stenger (8)	222	9,872
Robert Hearn	266	4,492
Jim Lloyd (1)	900	33,044

As always, Bob's code is well commented and fun to read. In addition to studying it for efficiency tips (and interesting macro definitions) you can bet that I'll be watching the moves it makes to try and learn how to put my cube back to its initial state. Maybe someday I'll be able to solve any cube in the theoretical 22 or 23 moves. Right, and maybe monkeys will fly out of my butt, too...

SOLVERUBIKSCUBE

Copyright (c) 1994 J Robert Boonstra

This source has been edited for length. The entire source is available at the usual online sites. Please see page 2 for details.

Problem Statement

Solve Rubiks cube, given an initial state by a call to MikeCubeToRubiksCube. Provide access to solution progress via RubiksCubeToMikeCube. Return 1 when cube is solved, 0 after an intermediate move, and -1 if the cube is unsolvable.

Background

Although it hasn't been proven, it is believed that "God's algorithm" for solving the cube requires something like 22 or 23 moves in the worst case. Back in the 1970s, when the cube was introduced, Singmaster and Thistlethwaite published

BBEdit 3.1



Not Your Father's Text Editor.

We've taken the accelerated for PowerPC™ text editor that you have come to rely on, put it on CD-ROM and added new features that we know your father would have wanted.

- Integrated support for Metrowerks CodeWarrior.
- Automatic text wrapping without those grungy carriage returns.
- The integrated PopupFuncs support now recognizes Rez, Fortran and 68K assembler source files.
- Electronic documentation with interactive browser, so the information you need is always at your fingertips.
- Tools from Bare Bones Software's freeware collection.
- BBEdition freeware and shareware extensions.
- The BBEdition Extension Developer's Kit.

Take some really cool software for a spin.

On top of all of that, the CD includes demos, updates, special offers and information from industry leaders, including: Aladdin Systems, Language Systems, MacTech Magazine, Mathemæsthetics, MindVision Software, Onyx Technology and Symantec.

How do I get mine?

Look in the Mail Order Store section of this issue for info about ordering BBEdition 3.1 or give us a call.

For a free demo disk, send us e-mail:

Internet: bbsw@netcom.com

CompuServe: 73051,3255

Applelink: BARE.BONES

E-World: BareBones

BBEdit 3.1, It Doesn't Suck.



We also have "It doesn't suck" t-shirts that don't either. Call us to order. © 1994 Bare Bones Software, Inc. P.O. Box 108 Bedford, MA 01760 voice: 508.651.3561 fax: 508.651.7584. Our lawyers insist we say: BBEdition, PopupFuncs and our spiffy logo are trademarks of Bare Bones Software, Inc. Mac and the Mac OS logo are trademarks of Apple Computer, Inc., used under license. PowerPC is a trademark of International Business Machines Corporation. All other trademarks and registered trademarks are properties of their respective holders.

POSITIONS WANTED

Available Immediately

1-800-736-3577

Expert 4D Programmers Less than 50¢ per hour!

More than 1,000 hours of development went into 4D Toolkit 2.0. With a one time price of \$395.00*, it's like hiring a crack team of 4D programmers at less than 50¢ per hour. Call 1-800-736-3577 to order your copy, or to receive a free demo.

4D TOOLKIT™

OPtions Computer Consulting
228 Bleecker Street #19
New York, NY 10014
TEL: 212-645-3577
FAX: 212-633-0336

*upgrade pricing available

solutions that solve the cube in ~52 moves, and that result has probably been improved upon since then. (These numbers may count half-turns as a single move instead of two quarter-turn moves – some people prefer to count that way.) For those interested in the cube, there is an active mailing list – send mail to cube-lovers-request@ai.mit.edu for more info. In the event that Singmaster, Thistlethwaite, God, or an avid cube-lover doesn't enter the challenge, we offer this solution.

Solution strategy

This solution is based on the now out-of-print book by Don Taylor, entitled *Mastering Rubik's Cube*, and sold at the time for the princely sum of \$1.95. While not in any way optimal, the solution in the book has the advantage of being (relatively) easy to remember.

We solve the cube using the following steps:

1. Solve the edge cubes in the top layer.
2. Solve the corner cubes in the top layer.
3. Solve the (edge) cubes in the middle layer.
4. Move the bottom layer corner cubes to the correct position.
5. Orient the bottom layer corner cubes correctly.
6. Move the bottom layer edge cubes to the correct position.
7. Orient the bottom layer edge cubes correctly.

This solution trades a large amount of space (code) for

speed. The operators that transform the cube are coded as macros, rather than as subroutines. (These macros were generated by an auxiliary program.)

rubik.c

```
#pragma options(honor_register,assign_registers)
#include "rubik.h"
#include "transform.h"

char *theMoveP; /* pointer to stored moves */
char *lastMoveP; /* pointer to final move */
short firstTime; /* set to 1 by MikeCubeToRubiksCube */

int SolveRubiksCube(register RubiksCube *rub)
{
    register unsigned short ch;
    if (firstTime) {
        // First time through, check to see if the cube is legal. Check for existence of the
        // required corners/edges. (We also could check the twist on the corners and the flip
        // parity on the edges to ensure that the cube is solvable, but I never got that working.)
        if (!LegalCube(rub)) return (-1);
        // Find complete solution on first call. Play back on subsequent calls. Initial solution
        // split into subroutine calls to deal with Symantec C limit on code in one file.
        SolveTopEdgesFR(rub);
        SolveTopEdgesLB(rub);
        if (!SolveTopCorners(rub)) return (-1);
        if (!SolveMiddleLayer(rub)) return (-1);
        if (!SolveBottomCorners(rub)) return (-1);
        if (!SolveBottomEdges(rub)) return (-1);
        firstTime = 0;

        // Restore the initial cube state so that we can play back the moves one at a time.
        lastMoveP = theMoveP;
        theMoveP = rub->theMove;
        {
            register long *p=(long *)&rub->cube[0][0];
            register ct=16*8/sizeof(long);
            do {
                *p = *(p+ 16*8/sizeof(long));
                ++p;
            } while (--ct);
        }
        ch = *theMoveP;
        switch (ch) {
            case U: U1move; break;
            case F: F1move; break;
            case L: L1move; break;
            case D: D1move; break;
            case B: B1move; break;
            case R: R1move; break;
            case u: U3move; break;
            case f: F3move; break;
            case l: L3move; break;
            case d: D3move; break;
            case b: B3move; break;
            case r: R3move; break;
        }
        return (lastMoveP == ++theMoveP);
    }

#define CornerVal(X,Y,Z) \
    ((1<(X##Y##Z##_##X)) | (1<(X##Y##Z##_##Y)) | \
     (1<(X##Y##Z##_##Z)))

#define EdgeVal(X,Z) \
    ((1<(X##Z##_##X)) | (1<(X##Z##_##Z)))

#define crn(a,b,c) ((1<a) | (1<b) | (1<c))
#define edg(a,b) ((1<a) | (1<b))

static Boolean LegalCube(RubiksCube *rub)
{
    char cubeValues[20];
    register long whichCubes=0;
    register char *valP;
    register short count,theVal;
```



```
// Make certain all the necessary corner cubes are there.
```

```
valP = cubeValues;
*valP++ = CornerVal(U,L,F);
*valP++ = CornerVal(U,R,F);
*valP++ = CornerVal(U,L,B);
*valP++ = CornerVal(U,R,B);
*valP++ = CornerVal(D,L,F);
*valP++ = CornerVal(D,R,F);
*valP++ = CornerVal(D,L,B);
*valP++ = CornerVal(D,R,B);
```

```
valP = cubeValues;
count=8;
whichCubes=0;
do {
    theVal = *valP++;
    if (theVal == crn(U,L,F)) {whichCubes|=0x01; continue;}
    if (theVal == crn(U,R,F)) {whichCubes|=0x02; continue;}
    if (theVal == crn(U,L,B)) {whichCubes|=0x04; continue;}
    if (theVal == crn(U,R,B)) {whichCubes|=0x08; continue;}
    if (theVal == crn(D,L,F)) {whichCubes|=0x10; continue;}
    if (theVal == crn(D,R,F)) {whichCubes|=0x20; continue;}
    if (theVal == crn(D,L,B)) {whichCubes|=0x40; continue;}
    if (theVal == crn(D,R,B)) {whichCubes|=0x80; continue;}
    return false;
} while (--count);
if (whichCubes != 0xFF) return false;
```

```
// Make certain all the necessary edge cubes are there.
```

```
valP = cubeValues+8;
*valP++ = EdgeVal(U,L); *valP++ = EdgeVal(U,R);
*valP++ = EdgeVal(D,L); *valP++ = EdgeVal(D,R);
*valP++ = EdgeVal(U,F); *valP++ = EdgeVal(U,B);
*valP++ = EdgeVal(D,F); *valP++ = EdgeVal(D,B);
*valP++ = EdgeVal(L,F); *valP++ = EdgeVal(L,B);
*valP++ = EdgeVal(R,F); *valP++ = EdgeVal(R,B);
```

```
valP = cubeValues+8;
count=12;
whichCubes=0;
do {
    theVal = *valP++;
    if (theVal == edg(U,L)) {whichCubes|=0x0001; continue;}
    if (theVal == edg(U,R)) {whichCubes|=0x0002; continue;}
    if (theVal == edg(D,L)) {whichCubes|=0x0004; continue;}
    if (theVal == edg(D,R)) {whichCubes|=0x0008; continue;}
    if (theVal == edg(U,F)) {whichCubes|=0x0010; continue;}
    if (theVal == edg(U,B)) {whichCubes|=0x0020; continue;}
    if (theVal == edg(D,F)) {whichCubes|=0x0040; continue;}
    if (theVal == edg(D,B)) {whichCubes|=0x0080; continue;}
    if (theVal == edg(L,F)) {whichCubes|=0x0100; continue;}
    if (theVal == edg(L,B)) {whichCubes|=0x0200; continue;}
    if (theVal == edg(R,F)) {whichCubes|=0x0400; continue;}
    if (theVal == edg(R,B)) {whichCubes|=0x0800; continue;}
    return false;
} while (--count);
if (whichCubes != 0xFFFF) return false;

return (true);
}
```

rubik.h

TYPEDEFS and DEFINES

```
typedef struct CubeSide {
    char littleSquare[3][3];
} CubeSide;
```

```
typedef struct MikeCube {
    CubeSide face[6];
} MikeCube;
```

```
// face ordering in MikeCube
```

```
enum {kTop=0, kLeft, kFront, kRight, kBottom, kBack};
```

```
typedef struct RubiksCube {
    char cubie[16][8];
    char origCube[16][8];
    char theMove[512];
} RubiksCube, *RubiksCubePtr;
```

Relational Database C/C++ Class Libraries

End Tool Tyranny

On-Time, On-Budget Product Development

Free yourself from the tyranny of complex database "solutions" and take back the control you need to complete your product on time and on budget.

Our multiplatform relational database class library, **RC/21**, offers astonishingly straightforward development discipline with extraordinary features not found in any other commercial database product. License fees are reasonable and affordable. Rational class design plus source availability provide flexibility and ample opportunity for customization. ODBC support leverages your development with numerous 3rd-party end-user tools. We offer toll-free product support and custom development by the **RC/21** developers.

Call 1-800-822-4437 To Order

- > \$149 full-featured Developer's Kit (single-CPU)
- > \$495 full single-user source code
- > \$495 Distribution License Pack
- > \$49 ODBC Driver
- > \$Call Client/Server Kits

Get Control Now!

**Vermont
Database
Corporation**

400 Upper Hollow Hill Road
Stowe, VT 05672-4510 USA
1-802-253-4437
1-802-253-4146 FAX
70334.3705@compuserve.com

Put Us On Your Team!

```
// face ordering in RubiksCube
```

```
enum { F=0,L,R,B,U,D,f,l,r,b,u,d};
```

```
// Macros Front(x) give access to individual cubies on the Front face.
```

```
// Similarly for other faces.
```

```
#define Front(x) rub->cubie[x][0]
#define Left(x) rub->cubie[x][1]
#define Right(x) rub->cubie[x][2]
#define Back(x) rub->cubie[x][3]
#define Up(x) rub->cubie[x][4]
#define Down(x) rub->cubie[x][5]
```

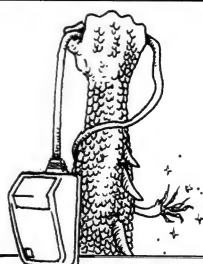
```
// Set up symbols to represent individual cubie faces
```

```
#define UL_F Front(0)
#define UF_F Front(1)
#define UR_F Front(2)
#define LF_F Front(3)
#define RF_F Front(5)
#define DL_F Front(6)
#define DF_F Front(7)
#define DR_F Front(8)
```

```
#define UL_B Left(0)
#define UL_L Left(1)
#define UL_F Left(2)
#define LB_L Left(3)
#define LF_L Left(5)
#define DL_B Left(6)
#define DL_L Left(7)
#define DL_F Left(8)
```

```
#define UR_F Right(0)
#define UR_R Right(1)
#define UR_B Right(2)
#define RF_R Right(3)
#define RB_R Right(5)
```


Green Dragon Creations



CUSTOM SOFTWARE DEVELOPMENT

Custom software development at reasonable rates. Past products include: Questick & Quepad ADB driver and control panel (published by MicroQue), Spectre VR (published by Velocity), and PowerFTP (published by Opensoft). We specialize in Macintosh software development and home game system software development.

Do you want to control X-10 devices from your Macintosh Application?

Control X-10 Devices from your Macintosh Applications! Use our library and it's easy! Royalty free distribution! Introductory price is \$50. Source code available at additional cost.

Green Dragon Creations, Inc.

412 Kimmons Avenue, Water Valley, Ms. 38965

1-601-473-4225

AppleLink: GREEN.DRAGON

Internet:howard_shere@greendragon.com

SOFTWARE ON THE EDGE

```
#define DRF_R Right(6)
#define DR_R Right(7)
#define DRB_R Right(8)
```

```
#define DLF_D Down(0)
#define DF_D Down(1)
#define DRF_D Down(2)
#define DL_D Down(3)
#define DR_D Down(5)
#define DLB_D Down(6)
#define DB_D Down(7)
#define DRB_D Down(8)
```

```
#define DLB_B Back(0)
#define DB_B Back(1)
#define DRB_B Back(2)
#define LB_B Back(3)
#define RB_B Back(5)
#define ULB_B Back(6)
#define UB_B Back(7)
#define URB_B Back(8)
```

```
#define ULB_U Up(0)
#define UB_U Up(1)
#define URB_U Up(2)
#define UL_U Up(3)
#define UR_U Up(5)
#define ULF_U Up(6)
#define UF_U Up(7)
#define URF_U Up(8)
```

```
/*
Macro M(x) records the individual turns in the transformation for playback during
subsequent calls to SolveRubiksCube.
*/
```

```
#define M(x) *theMoveP++ = x;
#define Rot2(a,b) \
```

```
(register char tmp; tmp=a; a=b; b=tmp;)
#define Rot3(a,b,c) \
(register char tmp; tmp=a; a=b; b=c; c=tmp;)
#define Rot4(a,b,c,d) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=tmp;)
#define Rot5(a,b,c,d,e) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=tmp;)
#define Rot6(a,b,c,d,e,f) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=tmp;)
#define Rot7(a,b,c,d,e,f,g) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=g; g=tmp;)
#define Rot8(a,b,c,d,e,f,g,h) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=g; g=h; h=tmp;)
#define Rot9(a,b,c,d,e,f,g,h,i) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=g; g=h; h=i; \
i=tmp;)
#define Rot10(a,b,c,d,e,f,g,h,i,j) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=g; g=h; h=i; \
i=j; j=tmp;)
#define Rot12(a,b,c,d,e,f,g,h,i,j,k,l) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=g; g=h; h=i; \
i=j; j=k; k=l; l=tmp;)
#define Rot15(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=g; g=h; h=i; \
i=j; j=k; k=l; l=m; m=n; n=o; o=tmp;)
#define Rot16(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p) \
(register char tmp; tmp=a; a=b; b=c; c=d; d=e; e=f; f=g; g=h; h=i; \
i=j; j=k; k=l; l=m; m=n; n=o; o=p; p=tmp;)
#define CornerEquals(X,Y,Z,a,b,c) \
((X##Y##Z##_##X==a | X##Y##Z##_##X==b | X##Y##Z##_##X==c) && \
(X##Y##Z##_##Y==a | X##Y##Z##_##Y==b | X##Y##Z##_##Y==c) && \
(X##Y##Z##_##Z==a | X##Y##Z##_##Z==b | X##Y##Z##_##Z==c))
#define CornerCorrect(X,Y,Z,a,b,c) \
((X##Y##Z##_##X==X | X##Y##Z##_##X==Y | X##Y##Z##_##X==Z) && \
(X##Y##Z##_##Y==X | X##Y##Z##_##Y==Y | X##Y##Z##_##Y==Z) && \
(X##Y##Z##_##Z==X | X##Y##Z##_##Z==Y | X##Y##Z##_##Z==Z))
```

```
int SolveRubiksCube(RubiksCube *cubePtr);
int FindSolution(void);
```

PROTOTYPES

```
void MikeCubeToRubiksCube(MikeCube *mikePtr,
RubiksCube *rubikPtr);
void RubiksCubeToMikeCube(RubiksCube *rubikPtr,
MikeCube *mikePtr);
void SolveTopEdgesFR(RubiksCube *rubPtr);
void SolveTopEdgesLB(RubiksCube *rubPtr);
Boolean SolveTopCorners(RubiksCube *rubPtr);
Boolean SolveMiddleLayer(RubiksCube *rubPtr);
Boolean SolveBottomCorners(RubiksCube *rubPtr);
Boolean SolveBottomEdges(RubiksCube *rubPtr);
Boolean LegalCube(RubiksCube *rubPtr);
```

```
extern char *theMoveP; /* pointer to stored moves */
extern short firstTime;
```

convert.c

```
#include "rubik.h"

/* mapping of MikeCube faces to RubiksCube faces */
char rubikFaceOrder[] = {F,L,R,B,U,D};
char mikeToRubik[] = {U,L,F,R,D,B};
char rubikToMike[] = {kFront,kLeft,kRight,kBack,kTop,kBottom};
char mikeColorToRubik[6];
char rubikColorToMike[6];
```

MikeCubeToRubiksCube

```
void MikeCubeToRubiksCube(MikeCube *mikePtr,
RubiksCube *rubikPtr)
{
short f,r,c;
theMoveP = rubikPtr->theMove;
firstTime = 1;
for (f=0; f<6; ++f) {
for (r=0; r<3; ++r) {
```



```

    for (c=0; c<3; ++c) {
/* delete s,d if numbering is corrected */
        short s=r,d=c;
        rubikPtr->cube[3*s+d][mikeToRubik[f]] =
            mikeToRubik[ mikePtr->face[f].littleSquare[r][c]];
    }
}

for (f=0; f<6; ++f) {
    char theColor;
    theColor = rubikFaceOrder[rubikPtr->cube[4][f]];
    rubikColorToMike[f] = theColor;
    mikeColorToRubik[theColor] = f;
}

for (f=0; f<6; ++f)
    for (r=0; r<3; ++r)
        for (c=0; c<3; ++c) {
            char *p = &rubikPtr->cube [3*r+c][f];
            *p = mikeColorToRubik[*p];
        }

for (f=0; f<6; ++f)
    for (r=0; r<3; ++r)
        for (c=0; c<3; ++c)
            rubikPtr->origCube[3*r+c][f] =
                rubikPtr->cube[3*r+c][f];
//PrintCube(rubikPtr,&oldRub,');
}

```

RubiksCubeToMikeCube

```

void RubiksCubeToMikeCube(RubiksCube *rubikPtr,
                           MikeCube *mikePtr)
{
    short f,r,c;
    for (f=0; f<6; ++f) {
        for (r=0; r<3; ++r) {
            for (c=0; c<3; ++c) {
                short s=r,d=c;

                mikePtr->face[f].littleSquare[r][c] =
                    rubikToMike[ rubikColorToMike[
                        rubikPtr->cube[3*s+d][mikeToRubik[f]] ] ];
            }
        }
    }
}

```

SolveTopEdgesFront-Right.c

```

#pragma options(honor_register,assign_registers)
#include "rubik.h"
#include "transform.h"

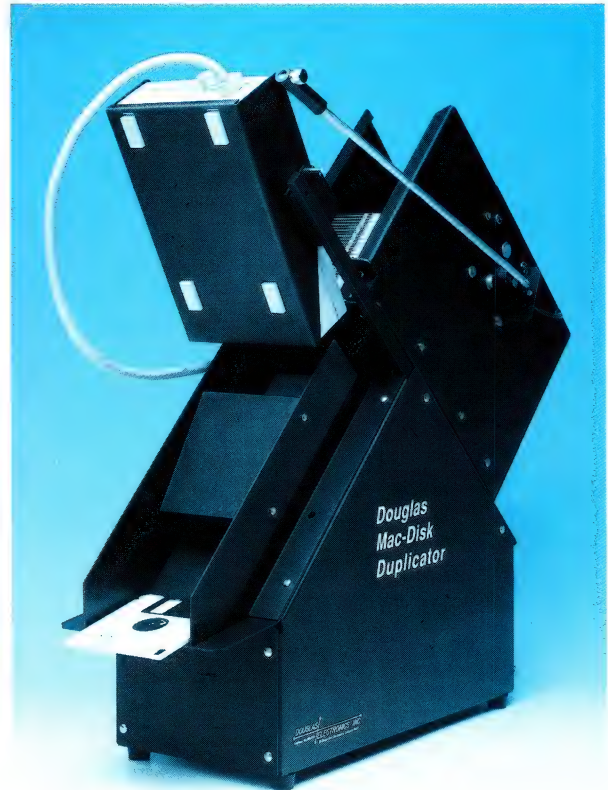
void SolveTopEdgesFR(register RubiksCube *rub)
{
    // STEP 1: Put the edge cubes in the top layer into the proper position and orientation.
    // Loop until all are correct.

    if (UF_U == U && UF_F == F) {
/* leave edge in correct position */;
    } else if (UF_U == F && UF_F == U) { F2D1R1F3R3;
    } else if (UR_U == F && UR_R == U) { R3F3;
    } else if (UB_U == F && UB_B == U) { B2D3R1F3R3;
    } else if (UL_U == F && UL_L == U) { L1F1;
    } else if (UR_U == U && UR_R == F) { R2D3F2;
    } else if (UB_U == U && UB_B == F) { B2D2F2;
    } else if (UL_U == U && UL_L == F) { L2D1F2;
    } else if (RF_F == F && RF_R == U) { F3;
    } else if (RB_R == F && RB_B == U) { B3D3B1R1F3R3;
    } else if (LB_B == F && LB_L == U) { B1D2B3F2;
    } else if (LF_L == F && LF_F == U) { L1D1L3F2;
    } else if (RF_F == U && RF_R == F) { F1D1R1F3R3;
    } else if (RB_R == U && RB_B == F) { R2F3R2;
    } else if (LB_B == U && LB_L == F) { B1D3B3R1F3R3;
    } else if (LF_L == U && LF_F == F) { F1;
    } else if (DF_D == F && DF_F == U) { D1R1F3R3;
    } else if (DR_D == F && DR_R == U) { R1F3R3;
    } else if (DB_D == F && DB_B == U) { D3R1F3R3;
    } else if (DL_D == F && DL_L == U) { L3F1L1;
    }
}

```

Douglas MacDisk Duplicator

The low-cost solution to disk duplication!



The simple, low cost, no frills MacDisk Duplicator is the answer to your software duplication needs. No other duplicator can match its ease of use and efficiency for the price. All you need to operate the duplicator is an Apple Macintosh computer. From there, you simply plug the duplicator's disk drive into the Macintosh, load the diskettes into the input hopper and get back to your other work. A short while later, your disks will be ready for labeling and distribution. Disk duplication couldn't be any easier or more economical than with the MacDisk Duplicator from Douglas Electronics. The MacDisk Duplicator runs on any Macintosh with an external floppy port.

Pricing:

\$1,500 without disk drive

\$1,850 with FDHD disk drive

For More Information:



**DOUGLAS
ELECTRONICS
INC.**

2777 Alvarado Street
[510] 483-8770

San Leandro, California 94577
FAX 483-6453

All **TEXT** is not created equal.

Is your application taking *cut & paste* a little too literally when it creates text? Are you tired of living in a monotype world? **PAIGE™**, our text & page layout programming library, is the ultimate cross-platform solution.

PAIGE provides the most sophisticated features/functions in the business. These include:

- **Stylized Text**
- **Shapes & Containers**
- **Text Wrapping**
- **Embedded Objects**
- **Hypertext Links**
- **Virtual Memory**
- **Style Sheet Support**
- **Multi-Level "Undo"**
- **Royalty Free**

So why should you join the **PAIGE** revolution? TIME. Most programmers don't have time to reinvent the wheel.

PAIGE was designed using no global variables and machine specific code has been isolated into two small source files. This strategy allows you to move your application to other operating systems or platforms by changing only platform specific code while maintaining full data and application compatibility.

Join the hundreds of major software publishers using **PAIGE** as their total text solution. For complete technical/pricing summary contact **DataPak Software** at 800-327-6703 or 206-573-9155.

Macintosh • Power Macintosh • Windows

```
    } else if (DF_D == U && DF_F == F) { F2;
    } else if (DR_D == U && DR_R == F) { D3F2;
    } else if (DB_D == U && DB_B == F) { D2F2;
    } else if (DL_D == U && DL_L == F) { D1F2;
    }

//Find the top-right edge cube, and move it into the proper position.
if (UR_U == U && UR_R == R) {
    /* leave edge in correct position */;
    } else if (UF_U == R && UF_F == U) { F1R1;
    } else if (UR_U == R && UR_R == U) { R2D1B1R3B3;
    } else if (UB_U == R && UB_B == U) { B3R3;
    } else if (UL_U == R && UL_L == U) { L2D3B1R3B3;
    } else if (UF_U == U && UF_F == R) { F2D1R2;
    } else if (UB_U == U && UB_B == R) { B2D3R2;
    } else if (UL_U == U && UL_L == R) { L2D2R2;
    } else if (RF_F == R && RF_R == U) { F1D1F3R2;
    } else if (RB_R == R && RB_B == U) { R3;
    } else if (LB_B == R && LB_L == U) { L3D3L1B1R3B3;
    } else if (LF_L == R && LF_F == U) { L1D2L3R2;
    } else if (RF_F == U && RF_R == R) { R1;
    } else if (RB_R == U && RB_B == R) { R1D1B1R3B3;
    } else if (LB_B == U && LB_L == R) { B2R3B2;
    } else if (LF_L == U && LF_F == R) { L1D3L3B1R3B3;
    } else if (DF_D == R && DF_F == U) { F3R1F1;
    } else if (DR_D == R && DR_R == U) { D1B1R3B3;
    } else if (DB_D == R && DB_B == U) { B1R3B3;
    } else if (DL_D == R && DL_L == U) { D3B1R3B3;
    } else if (DF_D == U && DF_F == R) { D1R2;
    } else if (DR_D == U && DR_R == R) { R2;
    } else if (DB_D == U && DB_B == R) { D3R2;
    } else if (DL_D == U && DL_L == R) { D2R2;
    }
}
```

SolveTopEdgesLeft-Back.c

```
#pragma options(honor_register,assign_registers)
```

```
#include "rubik.h"
#include "transform.h"

void SolveTopEdgesLB(register RubiksCube *rub)
{
    // Find the top-back edge cube, and move it into the proper position.
    if (UB_U == U && UB_B == B) { /* correct as is */
    } else if (UF_U == B && UF_F == U) { F2D3L1B3L3;
    } else if (UR_U == B && UR_R == U) { R1B1;
    } else if (UB_U == B && UB_B == U) { B2D1L1B3L3;
    } else if (UL_U == B && UL_L == U) { L3B3;
    } else if (RF_F == B && RF_R == U) { F1D2F3B2;
    } else if (RB_R == B && RB_B == U) { R1D1R3B2;
    } else if (LB_B == B && LB_L == U) { B3;
    } else if (LF_L == B && LF_F == U) { F3D3F1L1B3L3;
    } else if (DF_D == B && DF_F == U) { D3L1B3L3;
    } else if (DR_D == B && DR_R == U) { R3B1R1;
    } else if (DB_D == B && DB_B == U) { D1L1B3L3;
    } else if (DL_D == B && DL_L == U) { L1B3L3;
    } else if (UF_U == U && UF_F == B) { F2D2B2;
    } else if (UR_U == U && UR_R == B) { R2D1B2;
    } else if (UL_U == U && UL_L == B) { L2D3B2;
    } else if (RF_F == U && RF_R == B) { F1D3F3L1B3L3;
    } else if (RB_R == U && RB_B == B) { B1;
    } else if (LB_B == U && LB_L == B) { B1D1L1B3L3;
    } else if (LF_L == U && LF_F == B) { L2B3L2;
    } else if (DF_D == U && DF_F == B) { D2B2;
    } else if (DR_D == U && DR_R == B) { D1B2;
    } else if (DB_D == U && DB_B == B) { B2;
    } else if (DL_D == U && DL_L == B) { D3B2;
    }
}
```

```
// Find the top-left edge cube, and move it into the proper position.
if (UL_U == U && UL_L == L) { /* correct as is */
} else if (UF_U == L && UF_F == U) { F3L3;
} else if (UR_U == L && UR_R == U) { R2D3F1L3F3;
} else if (UB_U == L && UB_B == U) { B1L1;
} else if (UL_U == L && UL_L == U) { L2D1F1L3F3;
} else if (UF_U == U && UF_F == L) { F2D3L2;
} else if (UR_U == U && UR_R == L) { R2D2L2;
} else if (UB_U == U && UB_B == L) { B2D1L2;
} else if (RF_F == L && RF_R == U) { R3D3R1F1L3F3;
} else if (RB_R == L && RB_B == U) { R1D2R3L2;
} else if (LB_B == L && LB_L == U) { B1D1B3L2;
} else if (LF_L == L && LF_F == U) { L3;
} else if (RF_F == U && RF_R == L) { F2L3F2;
} else if (RB_R == U && RB_B == L) { R1D3R3F1L3F3;
} else if (LB_B == U && LB_L == L) { L1;
} else if (LF_L == U && LF_F == L) { L1D1F1L3F3;
} else if (DF_D == L && DF_F == U) { F1L3F3;
} else if (DR_D == L && DR_R == U) { D3F1L3F3;
} else if (DB_D == L && DB_B == U) { B3L1B1;
} else if (DL_D == L && DL_L == U) { D1F1L3F3;
} else if (DF_D == U && DF_F == L) { D3L2;
} else if (DR_D == U && DR_R == L) { D2L2;
} else if (DB_D == U && DB_B == L) { D1L2;
} else if (DL_D == U && DL_L == L) { L2;
}
}
```

SolveTopCorners

```
#pragma options(honor_register,assign_registers)
#include "rubik.h"
#include "transform.h"

Boolean SolveTopCorners(register RubiksCube *rub)
{
    short loopCount;
```

// STEP 2: Put the corner cubes in the top layer into the proper position and orientation. Loop until all are correct.

```
    loopCount=0;
    do {
        if (++loopCount>8) return false;
        if (DRF_F == U) {
            if (DRF_R == F) {
```



```

    if (DRF_D == R) {          goto URF1;
    } else if (DRF_D == L) { D3; goto ULF1;
    }
} else if (DRF_R == R) {
    if (DRF_D == F) {          goto URF1;
    } else if (DRF_D == B) { D1; goto URB1;
    }
} else if (DRF_R == B) {
    if (DRF_D == R) {          D1; goto URB1;
    } else if (DRF_D == L) { D2; goto ULB1;
    }
} else if (DRF_R == L) {
    if (DRF_D == F) {          D3; goto ULF1;
    } else if (DRF_D == B) { D2; goto ULB1;
    }
}
} else if (DRF_R == U) {
    if (DRF_F == F) {
        if (DRF_D == R) {          goto URF1;
        } else if (DRF_D == L) { D3; goto ULF1;
        }
    } else if (DRF_F == R) {
        if (DRF_D == F) {          goto URF1;
        } else if (DRF_D == B) { D1; goto URB1;
        }
    } else if (DRF_F == B) {
        if (DRF_D == R) {          D1; goto URB1;
        } else if (DRF_D == L) { D2; goto ULB1;
        }
    } else if (DRF_F == L) {
        if (DRF_D == F) {          D3; goto ULF1;
        } else if (DRF_D == B) { D2; goto ULB1;
        }
    }
} else if (DRF_D == U) {
    if (DRF_F == F) {
        if (DRF_R == R) {          goto URF1;
        } else if (DRF_R == L) { D3; goto ULF1;
        }
    } else if (DRF_F == R) {
        if (DRF_R == F) {          goto URF1;
        } else if (DRF_R == B) { D1; goto URB1;
        }
    } else if (DRF_F == B) {
        if (DRF_R == R) {          D1; goto URB1;
        } else if (DRF_R == L) { D2; goto ULB1;
        }
    } else if (DRF_F == L) {
        if (DRF_R == F) {          D3; goto ULF1;
        } else if (DRF_R == B) { D2; goto ULB1;
        }
    }
}
}
goto corner2;
URF1: if (DRF_F == U) { F1D1F3;
    } else if (DRF_R == U) { R3D3R1;
    } else if (DRF_D == U) { R3D1R1D2R3D3R1;
    }
    goto corner2;
URB1: if (DRB_R == U) { R1D1R3;
    } else if (DRB_B == U) { B3D3B1;
    } else if (DRB_D == U) { B3D1B1D2B3D3B1;
    }
    goto corner2;
ULB1: if (DLB_B == U) { B1D1B3;
    } else if (DLB_L == U) { L3D3L1;
    } else if (DLB_D == U) { L3D1L1D2L3D3L1;
    }
    goto corner2;
ULF1: if (DLF_L == U) { L1D1L3;
    } else if (DLF_F == U) { F3D3F1;
    } else if (DLF_D == U) { F3D1F1D2F3D3F1;
    }
}
corner2: ;

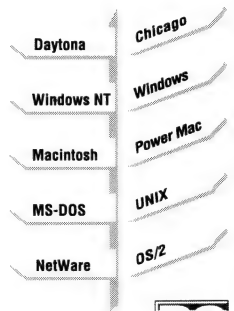
if (DRB_R == U) {
    if (DRB_B == F) {
        if (DRB_D == R) {          D3; goto URF2;
        } else if (DRB_D == L) { D2; goto ULF2;
        }
    } else if (DRB_B == R) {

```

WHETHER YOU'RE HEADED TO CHICAGO, DAYTONA, OR POINTS UNKNOWN, SUMMIT BASICSCRIPT HELPS YOU ARRIVE IN STYLE.

The BasicScript Toolkit makes it easy and economical to add an award-winning scripting language to your Macintosh, Power Macintosh, or Windows application.

- Easy-to-use APIs make it simple to integrate BasicScript into your application using C or C++.
- Compatible with the syntax of Microsoft Visual Basic and Visual Basic for Applications (VBA).
- Controls OLE Automation objects in Macintosh, Windows, and Windows NT applications.
- Extensibility architecture allows you to add your own keywords to the BasicScript language.
- Available for Macintosh, Power Macintosh, Windows 3.1, Windows 95 "Chicago", Windows NT 3.5 "Daytona", MS-DOS, OS/2, SunOS, Solaris 2.x, HP-UX, AIX, SCO UNIX, Ultrix, IRIX, and NetWare.
- Runtime can be redistributed by your customer without royalties.
- Script Editor with integrated debugger allows you to offer a complete Integrated Development Environment (IDE) for scripting.
- Free technical support, comprehensive documentation, and numerous code samples keep your engineering costs to a minimum.
- Licensed by Symantec, Delrina and other leading software companies. Named *PC Magazine* Editors' Choice among cross-application macro languages.
- Flexible licensing terms mean you can afford the finest scripting language on the market today.



SUMMIT

**CALL TODAY FOR YOUR FREE
EVALUATION COPY, 315-445-9000**

Summit Software Company Fax: 315-445-9567 Internet: info@summitsoft.com CompuServe: 71211,3504

```

    if (DRB_D == F) {          D3; goto URF2;
    } else if (DRB_D == B) { goto URB2;
    }
} else if (DRB_B == B) {
    if (DRB_D == R) {          goto URB2;
    } else if (DRB_D == L) { D1; goto ULB2;
    }
} else if (DRB_B == L) {
    if (DRB_D == F) {          D2; goto ULF2;
    } else if (DRB_D == B) { D1; goto ULB2;
    }
} else if (DRB_B == U) {
    if (DRB_R == F) {
        if (DRB_D == R) {          D3; goto URF2;
        } else if (DRB_D == L) { D2; goto ULF2;
        }
    } else if (DRB_R == R) {
        if (DRB_D == F) {          D3; goto URF2;
        } else if (DRB_D == B) { goto URB2;
        }
    } else if (DRB_R == B) {
        if (DRB_D == R) {          goto URB2;
        } else if (DRB_D == L) { D1; goto ULB2;
        }
    } else if (DRB_R == L) {
        if (DRB_D == F) {          D2; goto ULF2;
        } else if (DRB_D == B) { D1; goto ULB2;
        }
    }
} else if (DRB_D == U) {
    if (DRB_R == F) {
        if (DRB_B == R) {          D3; goto URF2;
        } else if (DRB_B == L) { D2; goto ULF2;
        }
    } else if (DRB_R == R) {
        if (DRB_B == F) {          D3; goto URF2;

```

Apprentice

Over 600 megabytes of up-to-date Mac-only source code and programmer utilities. Most of the source code is in CodeWarrior, Symantec, and MPW projects for C, C++, and Pascal. You'll find complete working examples of full-blown applications, games, control panels, extensions, utilities, and much more! \$35 including shipping. Add \$5 for shipping outside of the U.S. and Canada. VISA, MC, American Express, and Discover gladly accepted.



Celestin Company, 1152 Hastings Ave, Port Townsend, WA 98368
800 835 5514 • 206 385 3767 • 206 385 3586 fax
Internet: celestin@olympus.net • CompuServe: 71630,650

```

    } else if (DRB_B == B) { goto URB2;
    }
    } else if (DRB_R == B) {
        if (DRB_B == R) { goto URB2;
        } else if (DRB_B == L) { D1; goto ULB2;
        }
    } else if (DRB_R == L) {
        if (DRB_B == F) { D2; goto ULF2;
        } else if (DRB_B == B) { D1; goto ULB2;
        }
    }
    }
    goto corner3;
URF2: if (DRF_F == U) { F1D1F3;
    } else if (DRF_R == U) { R3D3R1;
    } else if (DRF_D == U) { R3D1R1D2R3D3R1;
    }
    goto corner3;
URB2: if (DRB_R == U) { R1D1R3;
    } else if (DRB_B == U) { B3D3B1;
    } else if (DRB_D == U) { B3D1B1D2B3D3B1;
    }
    goto corner3;
ULB2: if (DLB_B == U) { B1D1B3;
    } else if (DLB_L == U) { L3D3L1;
    } else if (DLB_D == U) { L3D1L1D2L3D3L1;
    }
    goto corner3;
ULF2: if (DLF_L == U) { L1D1L3;
    } else if (DLF_F == U) { F3D3F1;
    } else if (DLF_D == U) { F3D1F1D2F3D3F1;
    }
    corner3: ;
    if (DLB_B == U) {
        if (DLB_L == F) {
            if (DLB_D == R) { D2; goto URF3;
            } else if (DLB_D == L) { D1; goto ULF3;
            }
        }
    }

```

```

    } else if (DLB_L == R) {
        if (DLB_D == F) { D2; goto URF3;
        } else if (DLB_D == B) { D3; goto URB3;
        }
    } else if (DLB_L == B) {
        if (DLB_D == R) { D3; goto URB3;
        } else if (DLB_D == L) { goto ULB3;
        }
    } else if (DLB_L == L) {
        if (DLB_D == F) { D1; goto ULF3;
        } else if (DLB_D == B) { goto ULB3;
        }
    }
    } else if (DLB_L == U) {
        if (DLB_B == F) {
            if (DLB_D == R) { D2; goto URF3;
            } else if (DLB_D == L) { D1; goto ULF3;
            }
        } else if (DLB_B == R) {
            if (DLB_D == F) { D2; goto URF3;
            } else if (DLB_D == B) { D3; goto URB3;
            }
        } else if (DLB_B == B) {
            if (DLB_D == R) { D3; goto URB3;
            } else if (DLB_D == L) { goto ULB3;
            }
        } else if (DLB_B == L) {
            if (DLB_D == F) { D1; goto ULF3;
            } else if (DLB_D == B) { goto ULB3;
            }
        }
    }
    } else if (DLB_D == U) {
        if (DLB_B == F) {
            if (DLB_L == R) { D2; goto URF3;
            } else if (DLB_L == L) { D1; goto ULF3;
            }
        } else if (DLB_B == R) {
            if (DLB_L == F) { D2; goto URF3;
            } else if (DLB_L == B) { D3; goto URB3;
            }
        } else if (DLB_B == B) {
            if (DLB_L == R) { D3; goto URB3;
            } else if (DLB_L == L) { goto ULB3;
            }
        } else if (DLB_B == L) {
            if (DLB_L == F) { D1; goto ULF3;
            } else if (DLB_L == B) { goto ULB3;
            }
        }
    }
    }
    goto corner4;
URF3: if (DRF_F == U) { F1D1F3;
    } else if (DRF_R == U) { R3D3R1;
    } else if (DRF_D == U) { R3D1R1D2R3D3R1;
    }
    goto corner4;
URB3: if (DRB_R == U) { R1D1R3;
    } else if (DRB_B == U) { B3D3B1;
    } else if (DRB_D == U) { B3D1B1D2B3D3B1;
    }
    goto corner4;
ULB3: if (DLB_B == U) { B1D1B3;
    } else if (DLB_L == U) { L3D3L1;
    } else if (DLB_D == U) { L3D1L1D2L3D3L1;
    }
    goto corner4;
ULF3: if (DLF_L == U) { L1D1L3;
    } else if (DLF_F == U) { F3D3F1;
    } else if (DLF_D == U) { F3D1F1D2F3D3F1;
    }
    corner4: ;
    if (DLF_L == U) {
        if (DLF_F == F) {
            if (DLF_D == R) { D1; goto URF4;
            } else if (DLF_D == L) { goto ULF4;
            }
        } else if (DLF_F == R) {
            if (DLF_D == F) { D1; goto URF4;
            } else if (DLF_D == B) { D2; goto URB4;
            }
        } else if (DLF_F == B) {
            if (DLF_D == R) { D2; goto URB4;
            }
        }
    }

```



```

    } else if (DLF_D == L) { D3; goto ULB4;
    }
    } else if (DLF_F == L) {
    if (DLF_D == F) { goto ULF4;
    } else if (DLF_D == B) { D3; goto ULB4;
    }
    }
    } else if (DLF_F == U) {
    if (DLF_L == F) {
    if (DLF_D == R) { D1; goto URF4;
    } else if (DLF_D == L) { goto ULF4;
    }
    }
    } else if (DLF_L == R) {
    if (DLF_D == F) { D1; goto URF4;
    } else if (DLF_D == B) { D2; goto URB4;
    }
    }
    } else if (DLF_L == B) {
    if (DLF_D == R) { D2; goto URB4;
    } else if (DLF_D == L) { D3; goto ULB4;
    }
    }
    } else if (DLF_L == L) {
    if (DLF_D == F) { goto ULF4;
    } else if (DLF_D == B) { D3; goto ULB4;
    }
    }
    }
    } else if (DLF_D == U) {
    if (DLF_L == F) {
    if (DLF_F == R) { D1; goto URF4;
    } else if (DLF_F == L) { goto ULF4;
    }
    }
    } else if (DLF_L == R) {
    if (DLF_F == F) { D1; goto URF4;
    } else if (DLF_F == B) { D2; goto URB4;
    }
    }
    } else if (DLF_L == B) {
    if (DLF_F == R) { D2; goto URB4;
    } else if (DLF_F == L) { D3; goto ULB4;
    }
    }
    } else if (DLF_L == L) {
    if (DLF_F == F) { goto ULF4;
    } else if (DLF_F == B) { D3; goto ULB4;
    }
    }
    }
    }
    goto cornerDone;
URF4: if (DRF_F == U) { F1D1F3;
    } else if (DRF_R == U) { R3D3R1;
    } else if (DRF_D == U) { R3D1R1D2R3D3R1;
    }
    goto cornerDone;
URB4: if (DRB_R == U) { R1D1R3;
    } else if (DRB_B == U) { B3D3B1;
    } else if (DRB_D == U) { B3D1B1D2B3D3B1;
    }
    goto cornerDone;
ULB4: if (DLB_B == U) { B1D1B3;
    } else if (DLB_L == U) { L3D3L1;
    } else if (DLB_D == U) { L3D1L1D2L3D3L1;
    }
    goto cornerDone;
ULF4: if (DLF_L == U) { L1D1L3;
    } else if (DLF_F == U) { F3D3F1;
    } else if (DLF_D == U) { F3D1F1D2F3D3F1;
    }
    cornerDone;
//Exit if all 4 corner cubes in the top row are in the correct position and orientation.
if ((ULF_U==U && URF_U==U && ULB_U==U && URB_U==U &&
    ULF_F==F && URF_F==F && URF_R==R && URB_R==R &&
    ULB_B==B && URB_B==B && ULF_L==L && ULB_L==L)
    break;
//Move an incorrectly oriented corner cube in the top row down into the bottom row
if ((URF_U == U || URF_F == U || URF_R == U) &&
    (URF_U != U || URF_F != F || URF_R != R)) {
    R3D1R1;
} else if ((URB_U == U || URB_R == U || URB_B == U) &&
    (URB_U != U || URB_R != R || URB_B != B)) {
    B3D1B1;
} else if ((ULB_U == U || ULB_B == U || ULB_L == U) &&
    (ULB_U != U || ULB_B != B || ULB_L != L)) {
    L3D1L1;
} else if ((ULF_U == U || ULF_L == U || ULF_F == U) &&
    (ULF_U != U || ULF_L != L || ULF_F != F)) {
    F3D1F1;
} while (true);
return (true);
}

(SolveMiddleLayer.c)
#pragma options(honor_register,assign_registers)
#include "rubik.h"
#include "transform.h"

// Moves to transfer a cube to the middle face.


/* FDDLIDDf Up face matches Left center */
#define FDDLIDDf F1;D2;L1;D1;L3;D2;F3;
#define RDDDFDdr R1;D2;F1;D1;F3;D2;R3;
#define BDDDRdDb B1;D2;R1;D1;R3;D2;B3;
#define LDBDBdDdL L1;D2;B1;D1;B3;D2;L3;

Boolean SolveMiddleLayer(register RubiksCube *rub)
{
    short loopCount;
    loopCount=0;

    // STEP 3: Put the (edge) cubes in the middle layer into the proper position and
    // orientation. Loop until all are correct.

    do {
        if (++loopCount > 24) return false;
        if (DF_F != D && DF_D != D) { /* DF in wrong position*/
            if (DF_F == F && DF_D == R) { D3F1D2L1D1L3D2F3;
            } else if (DF_F == R && DF_D == F) { F1D2L1D1L3D2F3;
            } else if (DF_F == R && DF_D == B) { R1D2F1D1F3D2R3;
            } else if (DF_F == B && DF_D == R) { D1R1D2F1D1F3D2R3;
            } else if (DF_F == B && DF_D == L) { D1B1D2R1D1R3D2B3;
            }
        }
    } while (true);
}

```



Power Up with PowerPC!

Apple's Developer University has the right courses to jump start your PowerPC development efforts:

- ▶ Programmer's Introduction to RISC and PowerPC self-paced training
- ▶ PowerPC BootCamp Class

And, with over 19 courses, we offer the fastest way to get up to speed on the Macintosh, whether you're just getting started or seeking to understand Apple's newest development technologies.

For more information, contact the Apple Developer University Registrar by telephone at (408) 974-4897 or fax (408) 974-0544.

Developer University, Apple Computer, Inc. 1 Infinite Loop, MS 305-1TU, Cupertino, CA 95014

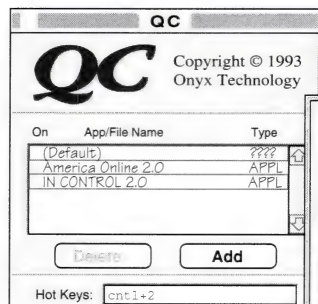


NOW SHIPPING

QC: the Macintosh Testing solution.

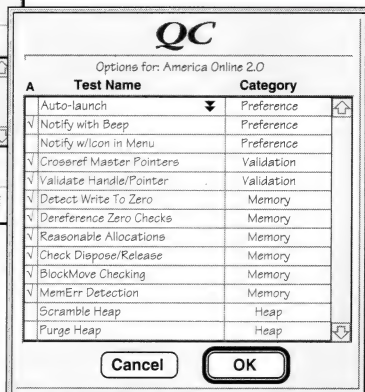
Subject your code to **brutal stress** conditions to **make it break** consistently.

Or use QC during the development cycle to casually detect block boundary overwrites, invalid BlockMoves, writes to location zero, and more... **saving countless hours of needless debugging.**



Features:

- Works with any program without modification - source not required
- Easy to use: just hit the hotkey.
- Fast heap scramble and purge
- Invalidates all free memory
- Detects runtime block overwrites
- Warns of DisposeHandle on resource and ReleaseResource on handles.
- Validates BlockMove destinations
- Sophisticated heap verification
- Powerful API for precision control



30 DAY
NO QUESTIONS
ASKED
MONEY BACK
GUARANTEE



ONYX TECHNOLOGY
7811 27th Ave W.
Bradenton, FL 34209

\$99 SPECIAL
INTRODUCTORY
PRICE

Tel: 813.796.7801 Fax: 813.792.5152 AOL: OnyxTech ALink: D2238 CIS: 70550,1377

```

} else if (DF_F == L && DF_D == B) { D2B1D2R1D1R3D2B3;
} else if (DF_F == L && DF_D == F) { D2L1D2B1D1B3D2L3;
} else if (DF_F == F && DF_D == L) { D3L1D2B1D1B3D2L3;
}
continue;
} else if (DR_R != D && DR_D != D) { /* DR in wrong pos */
if (DR_R == F && DR_D == R) { D2F1D2L1D1L3D2F3;
} else if (DR_R == R && DR_D == F) { D3F1D2L1D1L3D2F3;
} else if (DR_R == R && DR_D == B) { D3R1D2F1D1F3D2R3;
} else if (DR_R == B && DR_D == R) { R1D2F1D3F3D2R3;
} else if (DR_R == B && DR_D == L) { B1D2R1D1R3D2B3;
} else if (DR_R == L && DR_D == B) { D1B1D2R1D3R3D2B3;
} else if (DR_R == L && DR_D == F) { D1L1D2B1D1B3D2L3;
} else if (DR_R == F && DR_D == L) { D2L1D2B1D1B3D2L3;
}
continue;
} else if (DB_B != D && DB_D != D) { /* DB in wrong pos */
if (DB_B == F && DB_D == R) { D1F1D2L1D1L3D2F3;
} else if (DB_B == R && DB_D == F) { D2F1D2L1D1L3D2F3;
} else if (DB_B == R && DB_D == B) { D2R1D2F1D1F3D2R3;
} else if (DB_B == B && DB_D == R) { D3R1D2F1D1F3D2R3;
} else if (DB_B == B && DB_D == L) { D3B1D2R1D1R3D2B3;
} else if (DB_B == L && DB_D == B) { B1D2R1D3R3D2B3;
} else if (DB_B == L && DB_D == F) { D1L1D2B1D1B3D2L3;
} else if (DB_B == F && DB_D == L) { D1L1D2B1D1B3D2L3;
}
continue;
} else if (DL_L != D && DL_D != D) { /* DL in wrong pos */
if (DL_L == F && DL_D == R) { F1D2L1D1L3D2F3;
} else if (DL_L == R && DL_D == F) { D1F1D2L1D1L3D2F3;
} else if (DL_L == R && DL_D == B) { D1R1D2F1D1F3D2R3;
} else if (DL_L == B && DL_D == R) { D2R1D2F1D1F3D2R3;
} else if (DL_L == B && DL_D == L) { D2B1D2R1D1R3D2B3;
} else if (DL_L == L && DL_D == B) { D3B1D2R1D1R3D2B3;
} else if (DL_L == L && DL_D == F) { D3L1D2B1D1B3D2L3;
} else if (DL_L == F && DL_D == L) { L1D2B1D1B3D2L3;
}
continue;
}
}

```

```

//Exit if all edge cubes in the middle layer are in the correct position and orientation.

```

```

else if (LF_F == F && RF_F == F &&
RF_R == R && RB_R == R &&
LB_B == B && RB_B == B &&
LF_L == L && LB_L == L) {
break;
} else {

```

```

/*
All edges are not correct, but there are no edge cubes in the bottom layer that belong
in the middle layer. Need to move an incorrectly placed cube from the middle layer
into the bottom layer, so that the next loop can orient it correctly.
*/

```

```

if (RF_F != F || RF_R != R) { F1D2L1D1L3D2F3;
} else if (RB_R != R || RB_B != B) { R1D2F1D1F3D2R3;
} else if (LB_B != B || LB_L != L) { B1D2R1D1R3D2B3;
} else if (LF_L != L || LF_F != F) { L1D2B1D1B3D2L3;
}
continue;
} while(true);
return (true);
}

```

SolveBottomCorners.c

```

#pragma options(honor_register,assign_registers)
#include "rubik.h"
#include "transform.h"

```

```

Boolean SolveBottomCorners(register RubiksCube *rub)
{
short loopCount;

```

```

// STEP 4: Move corner cubes in bottom layer into position (but not necessarily the
// correct orientation)

```

```

if ( CornerEquals(D,L,B,D,R,B) ) { D3;
} else if ( CornerEquals(D,L,F,D,R,B) ) { D2;
} else if ( CornerEquals(D,R,F,D,R,B) ) { D1;
}

```

```

/*
Given that one corner (DRB) is in the correct position, move the other corners into
the correct position. There are (according to Taylor), 4 possibilities, clockwise
rotation of the three other corners, counterclockwise rotation, horizontal exchange (of
two), or diagonal exchange of two.
*/

```

```

if ( CornerCorrect(D,R,F) ) { /* DRF correct */
if ( CornerCorrect(D,L,F) ) { /* DLF correct */
;
} else {
R1D3L3D1R3D3L1D2; /* Exchange DLF and DLB */
}
} else {
if ( CornerCorrect(D,L,F) ) { /* Exchange DLB and DRF */
D1B1D1R1D3R3B3;
} else {
if ( CornerCorrect(D,L,B) ) { /* Exchange DLF and DRF */
B1D3F3D1B3D3F1D2;
} else {
if ( CornerEquals(D,L,B,D,L,F) ) {
L3D1R1D3L1D1R3D3; /* DLF<DLB<DRF */
} else {
D1R1D3L3D1R3D3L1; /* DLF<DRF<DLB */
}
}
}
}
}

```

```

//STEP 5: Twist corners in bottom layer.

```

```

loopCount=0;
do {
if (++loopCount > 16) return (false);
// At this point, all the corners in the bottom layer are in the correct positions, but
// perhaps not in the correct orientation.
if (DLF_F == F && DRF_R == R &&
DRB_B == B && DLB_L == L)
break;
}

```

```

/*
Not all of the corners are in the correct orientation. The cube has the property that
the "twist" of corner cubes sums to zero, meaning that we have one of the following
cases:

```


- 3 cubes needing a clockwise twist
 - 2 cubes needing a clockwise twist and 2 needing a counterclockwise twist
 - 1 cube needing a counterclockwise twist and 1 needing a clockwise twist
 - 3 cubes needing a counterclockwise twist
 The operators used in this cube solution twist one corner clockwise and one counterclockwise.

```

*/
    if (DLF_F == D) { /* DLF needs a clockwise twist */
        L3U1L1F1U1F3;

//Find a cube that needing a counterclockwise turn.

        if (DLB_B == D) {
LabD1F1U3F3L3U3L1D3:
            D1F1U3F3L3U3L1D3;
        } else if (DRB_R == D) {
            D2F1U3F3L3U3L1D2;
        } else if (DRF_F == D) {
LabD3F1U3F3L3U3L1D1:
            D3F1U3F3L3U3L1D1;
        } else { // No counterclockwise turn is needed, so we make one arbitrarily
            if (DLB_D != D) {
                goto LabD1F1U3F3L3U3L1D3;
            } else {
                goto LabD3F1U3F3L3U3L1D1;
            }
        }
    } else if (DRF_R == D) { /* DRF needs a clockwise twist */
        F3U1F1R1U1R3;
        if (DLF_L == D) {
LabD1R1U3R3F3U3F1D3:
            D1R1U3R3F3U3F1D3;
        } else if (DLB_B == D) {
            D2R1U3R3F3U3F1D2;
        } else if (DRB_R == D) {
LabD3R1U3R3F3U3F1D1:
            D3R1U3R3F3U3F1D1;
        } else { // No counterclockwise turn is needed, so we make one arbitrarily
            if (DLF_D != D) {
                goto LabD1R1U3R3F3U3F1D3;
            } else {
                goto LabD3R1U3R3F3U3F1D1;
            }
        }
    } else if (DRB_B == D) { /* DRB needs a clockwise twist */
        R3U1R1B1U1B3;
        if (DRF_F == D) {
LabD1B1U3B3R3U3R1D3:
            D1B1U3B3R3U3R1D3;
        } else if (DLF_L == D) {
            D2B1U3B3R3U3R1D2;
        } else if (DLB_B == D) {
LabD3B1U3B3R3U3R1D1:
            D3B1U3B3R3U3R1D1;
        } else { // No counterclockwise turn is needed, so we make one arbitrarily
            if (DRF_D != D) {
                goto LabD1B1U3B3R3U3R1D3;
            } else {
                goto LabD3B1U3B3R3U3R1D1;
            }
        }
    } else if (DLB_L == D) { /* DLB needs a clockwise twist */
        B3U1B1L1U1L3;
        if (DRB_R == D) {
LabD1L1U3L3B3U3B1D3:
            D1L1U3L3B3U3B1D3;
        } else if (DRF_F == D) {
            D2L1U3L3B3U3B1D2;
        } else if (DLF_L == D) {
LabD3L1U3L3B3U3B1D1:
            D3L1U3L3B3U3B1D1;
        } else { // No counterclockwise turn is needed, so we make one arbitrarily
            if (DRB_D != D) {
                goto LabD1L1U3L3B3U3B1D3;
            } else {
                goto LabD3L1U3L3B3U3B1D1;
            }
        }
    } else {
// There are no corner cubes that need a clockwise twist. So there must be 3 needing
// a counterclockwise twist. We twist one clockwise and one counterclockwise.

```

Programmer Training

MACINTOSH SEMINARS & CONSULTING

Richey Software Training provides professional, *customized* programming seminars and industry consulting.

Rich content & hands-on lab exercises shorten your learning curve. On-site training is convenient — your team eliminates expensive travel costs and consuming down-time.

“Professional training & consultancy that really hits the mark.”

Call today to find out how Richey Software Training delivers professional seminars and consulting nationwide.



Training Mac Programmers Since 1986

707-869-2836

AppleLink: RICHEYSOFT

INTERNET: 70413.2710@compuserve.com
 P.O. BOX 1809, GUERNEVILLE, CA 95446-1809

© 1994 Richey Software Training. All trademarks or registered trademarks are the property of their respective owners. Specifications subject to change.

MAC SEMINARS

- C & C++
- OOP
- MacApp
- PowerPC
- AppleScript
- MPW
- System 7
- Debugging
- SourceBug

```

if (DLF_F != F) {
    L3U1L1F1U1F3D1F1U3F3L3U3L1D3;
} else {
    F3U1F1R1U1R3D1R1U3R3F3U3F1D3;
}
} while(true);
return (true);
}

```

SolveBottomEdges.c

```

#pragma options(honor_register,assign_registers)
#include "rubik.h"
#include "transform.h"

```

```

Boolean SolveBottomEdges(register RubiksCube *rub)
{
    short loopCount;

```

// STEP 6: Move edge cubes in bottom layer into position.

```

if (DF_F == F || DF_D == F) { /* FD in pos */
    if (DL_L == R || DL_D == R) { /* BD->LD->RD->BD */
        B2D1R3L1B2R1L3D1B2;
    } else if (DR_R == L || DR_D == L) { /* BD->RD->LD->BD */
        B2D3R3L1B2R1L3D3B2;
    }
} else if (DL_L == L || DL_D == L) { /* LD in pos */
    if (DB_B == F || DB_D == F) { /* RD->BD->FD->RD */
        R2D1F3B1R2F1B3D1R2;
    } else if (DF_F == B || DF_D == B) { /* RD->FD->BD->RD */
        R2D3F3B1R2F1B3D3R2;
    }
} else if (DR_R == R || DR_D == R) { /* RD in pos */
    if (DF_F == B || DF_D == B) { /* LD->FD->BD->LD */

```

When you can't afford to slip, and the product just **has** to ship...

call
THE MAC GROUP
intense, focused debugging
aimed at helping you ship

800 SyncWait

796-2924

24 hours – we make housecalls

```
L2D1B3F1L2B1F3D1L2;
} else if (DB_B == F || DB_D == F) { /* LD->BD->FD->LD */
L2D3B3F1L2B1F3D3L2;
}
} else if (DB_B == B || DB_D == B) { /* BD in pos */
if (DR_R == L || DR_D == L) { /* FD->RD->LD->FD */
F2D1L3R1F2L1R3D1F2;
} else if (DL_L == R || DL_D == R) { /* FD->LD->RD->FD */
F2D3L3R1F2L1R3D3F2;
}
} else {
/* There are no edges in their proper place. */
if (DF_F == L || DF_D == L) {
F1L1D1L3D3F2R3D3R1D1F1;
} else if (DF_F == B || DF_D == B) {
R2L2U1R2L2D2R2L2U1R2L2;
} else if (DF_F == R || DF_D == R) {
R1F1D1F3D3R2B3D3B1D1R1;
}
}
}
```

//STEP 7: Flip edges in bottom layer.

```
loopCount = 0;
do {
if (++loopCount > 24) return false;
//At this point, all the edge cubes in the bottom layer are in the correct positions, but
//perhaps not in the correct orientation.
```

//Exit if all edge cubes have the proper orientation.

```
if (DF_F == F && DR_R == R &&
DB_B == B && DL_L == L)
break;
// At least one edge cubes does not have the proper orientation. The cube has the
// property that an even number of edge cubes need to be flipped.
```

```
if (DF_F == D) {
if (DL_L == D) {
F1D1U3R2D2U2L1D1L3U2D2R2U1D3F3D3;
} else if (DB_B == D) {
F1D1U3R2D2U2L1D1L3U2D2R2U1D3F3D2;
} else if (DR_R == D) {
F1D1U3R2D2U2L1D1L3U2D2R2U1D3F3D1;
}
} else if (DL_L == D) {
if (DB_B == D) {
L1D1U3F2D2U2B1D1B3U2D2F2U1D3L3D3;
} else if (DR_R == D) {
L1D1U3F2D2U2B1D1B3U2D2F2U1D3L3D2;
} else if (DF_F == D) {
L1D1U3F2D2U2B1D1B3U2D2F2U1D3L3D1;
}
}
```

```
} else if (DB_B == D) {
if (DR_R == D) {
B1D1U3L2D2U2R1D1R3U2D2L2U1D3B3D3;
} if (DF_F == D) {
B1D1U3L2D2U2R1D1R3U2D2L2U1D3B3D2;
} else if (DL_L == D) {
B1D1U3L2D2U2R1D1R3U2D2L2U1D3B3D1;
}
} else if (DR_R == D) {
if (DF_F == D) {
R1D1U3B2D2U2F1D1F3U2D2B2U1D3R3D3;
} else if (DL_L == D) {
R1D1U3B2D2U2F1D1F3U2D2B2U1D3R3D2;
} else if (DB_B == D) {
R1D1U3B2D2U2F1D1F3U2D2B2U1D3R3D1;
}
}
} while(true);
return (true);
}
```

transform.h

// F1move transforms the cube in response to a clockwise turn of the Front face.
// F3move represents a counter-clockwise turn. Similarly for the other faces.

```
#define F1move \
Rot4(ULF_U,DLF_L,DRF_D,URF_R); \
Rot4(ULF_L,DLF_D,DRF_R,URF_U); \
Rot4(ULF_F,DLF_F,DRF_R,URF_R); \
Rot4(UF_U,LF_L,DF_D,RF_R); \
Rot4(UF_F,LF_F,DF_F,RF_F);
...and so on...
```

//This file contains the permutations used to transform the cube during the calculation
//of the solution during the first call to SolveRubiksCube.

```
#define F1 F1move; M(F);
#define F3 F3move; M(f);
#define L1 L1move; M(L);
#define L3 L3move; M(l);
#define R1 R1move; M(R);
#define R3 R3move; M(r);
#define B1 B1move; M(B);
#define B3 B3move; M(b);
#define U1 U1move; M(U);
#define U3 U3move; M(u);
#define D1 D1move; M(D);
#define D3 D3move; M(d);
#define F2 \
Rot2(ULF_U,DRF_D); Rot2(ULF_L,DRF_R); \
```



```

Rot2(ULF_F,DRF_F); Rot2(DLF_D,URF_U); \
Rot2(DLF_L,URF_R); Rot2(DLF_F,URF_F); \
Rot2(ULF_U,DF_D); Rot2(ULF_F,DF_F); \
Rot2(LF_L,RF_R); Rot2(LF_F,RF_F); \
M(F);M(F);
#define L2 \
Rot2(ULF_U,DLB_D); Rot2(ULF_L,DLB_L); \
Rot2(ULF_F,DLB_B); Rot2(DLF_D,ULB_U); \
Rot2(DLF_L,ULB_L); Rot2(DLF_F,ULB_B); \
Rot2(UL_U,DL_D); Rot2(UL_L,DL_L); \
Rot2(LF_L,LB_L); Rot2(LF_F,LB_B); \
M(L);M(L);
#define R2 \
Rot2(URF_U,DRB_D); Rot2(URF_R,DRB_R); \
Rot2(URF_F,DRB_B); Rot2(DRF_D,URB_U); \
Rot2(DRF_R,URB_R); Rot2(DRF_F,URB_B); \
Rot2(UR_U,DR_D); Rot2(UR_R,DR_R); \
Rot2(RF_R,RB_R); Rot2(RF_F,RB_B); \
M(R);M(R);
and so on...

```

```

#define R3D1R1 \
Rot4(DLF_D,DLB_D,DRF_F,URF_F); \
Rot4(DLF_L,DLB_B,DRF_R,URF_U); \
Rot4(DLF_F,DLB_L,DRF_D,URF_R); \
Rot4(RF_R,DF_F,DL_L,DB_B); \
Rot4(RF_F,DF_D,DL_D,DB_D); \
M(r);M(D);M(R);
#define B3D1B1 \
Rot4(DLF_D,DRB_R,URB_R,DRF_D); \
Rot4(DLF_L,DRB_B,URB_U,DRF_F); \
Rot4(DLF_F,DRB_D,URB_B,DRF_R); \
Rot4(RB_R,DR_D,DF_D,DL_D); \
Rot4(RB_B,DR_R,DF_F,DL_L); \
M(b);M(D);M(B);
...much, much more of the same...

```

```

#define D3F1D2L1D1L3D2F3 \
Rot3(DLF_D,DLB_D,DRF_R); \
Rot3(DLF_L,DLB_B,DRF_D); \
Rot3(DLF_F,DLB_L,DRF_F); \
Rot4(RF_R,DF_D,RF_F,DF_F); \
Rot4(DR_D,DB_D,DR_R,DB_B); \
M(d);M(F);M(D);M(D);M(L);M(D);M(1);M(D);M(D);M(f);
#define F1D2L1D1L3D2F3 \
Rot4(DLF_D,DRB_D,DRF_D,DLB_B); \
Rot4(DLF_L,DRB_R,DRF_F,DLB_L); \
Rot4(DLF_F,DRB_B,DRF_R,DLB_D); \
Rot4(RF_R,DL_D,DB_D,DF_F); \
Rot4(RF_F,DL_L,DB_B,DF_D); \
M(F);M(D);M(D);M(L);M(D);M(1);M(D);M(D);M(f);
...much more of the same...

```

```

#define R1D3L3D1R3D3L1D2 \
Rot6(DLF_D,DLB_D,DLF_F,DLB_L,DLF_L,DLB_B); \
Rot3(DRF_D,DRF_R,DRF_F); \
Rot3(DRB_D,DRB_B,DRB_R); \
Rot4(DL_D,DB_D,DR_D,DF_D); \
Rot4(DL_L,DB_B,DR_R,DF_F); \
M(R);M(d);M(1);M(D);M(r);M(d);M(L);M(D);M(D);
#define D1B1D1R1D3R3B3 \
Rot3(DLF_D,DLF_F,DLF_L); \
Rot6(DRF_D,DLB_D,DRF_F,DLB_B,DRF_R,DLB_L); \
Rot4(DL_D,DB_D,DF_F,DR_D); \
Rot4(DL_L,DB_B,DF_D,DR_R); \
M(D);M(B);M(D);M(R);M(d);M(r);M(b);
...much more of the same...

```

```

#define L3U1L1F1U1F3 \
Rot3(ULF_U,ULF_L,ULF_F); \
Rot3(DLF_D,DLF_F,DLF_L); \
Rot9(URF_U,URB_U,ULB_U,URF_F,URB_R,ULB_B,URF_R,URB_B, \
ULB_L); \
Rot5(UL_U,UF_F,UR_R,LF_L,UB_U); \
Rot5(UL_L,UF_U,UR_U,LF_F,UB_B); \
M(1);M(U);M(L);M(F);M(U);M(F);
#define D1F1U3F3L3U3L1D3 \

```

NEW! Version 2.0 - Supports PPC & Fat Binary

PatchWorks™

Builds Updaters Without Programming

PatchWorks has many options, but only one function: to create updater applications for distribution via public channels (e.g., online services).

Before the advent of **PatchWorks**, creating an updater was a project in itself, one that consumed valuable programmer time which could more profitably be spent on revenue-producing projects.

With **PatchWorks**, you create an updater in minutes. Since there's no coding or scripting, no bugs are introduced. Just fill in a dialog, and **PatchWorks** does the rest!

Distribute updaters frequently to reflect maintenance releases, and watch your tech support and fulfillment costs fall dramatically.

Most important, your customers will know you care.

Features

- Works with apps, INITs, cdevs, fonts, drivers, etc.
- Now supports PPC, fat binary, & 4D apps
- Customizable user interface
- Updaters support multiple old versions
- Resource compression (diffing) produces small updaters
- Preserves personalization data (name, serial #, etc.)
- Updater distribution is **unrestricted & royalty-free**

Pricing: Begins at \$195. Call for more information.



SNA, Inc.
2200 NW Corporate Blvd.
Boca Raton, FL 33431
Tel (407) 241-0308 • FAX (407) 241-3195

Apple's Installer 4.0[®] ScriptGen Pro 2.0

Install your product using the familiar interface of Apple's Installer (the same program your customers use to install System software).

New in ScriptGen Pro 2.0

- **Installer 4.0 Support**
- **Developer Hooks**
- **Folder Copies**
- **Interface & Speed Improvements**

"Its intuitive interface makes creating Installer scripts blissfully simple and quick"

MacTech Magazine

Functional Demo Available:
AppleLink: Third Party Demos
AOL: Developers Forum
CIS: Developers Forum
eWorld

ScriptGen Pro: \$169

InstallerPack: \$219

**STEPUP
SOFTWARE**

7110 Glendora Avenue
Dallas, Texas 75230
214-360-9301
214-360-0127 fax

ScriptGen Pro is a trademark of StepUp Software. Apple is a registered trademark of Apple Computer, Inc.
* Requires additional licensing

Pascal → C++

Stuck with Pascal source code?

Want to move to C++?

OP2CPlus is a Macintosh tool for converting Object Pascal source code to C++ source code. It has already been used to translate hundreds of thousands of lines of Object Pascal to C++.

Ease your transition to OpenDoc or PowerPC

- ☐ Convert in days instead of months
- ☐ Generates C++ classes
- ☐ Works with MacApp 2 or 3
- ☐ Translates Think or MPW Pascal
- ☐ Full ANSI C source code included
- ☐ Just \$895

Graphic Magic Inc, 180 Seventh Ave, Suite 201
Santa Cruz CA 95062 Tel (408) 464 1949
Fax (408) 464 0731 AppleLink GRAPHICMAGIC

```
Rot3(ULF_U,ULF_F,ULF_L); \
Rot9(URF_U,ULB_L,URB_B,URF_R,ULB_B,URB_R,URF_F,ULB_U, \
URB_U); \
Rot3(DLB_D,DLB_B,DLB_L); \
Rot5(UL_U,UB_U,LF_L,UR_R,UF_F); \
Rot5(UL_L,UB_B,LF_F,UR_U,UF_U); \
M(D);M(F);M(u);M(f);M(l);M(u);M(L);M(d);
...much more of the same...
```

Edge exchanges

```
#define F1D1U3R2D2U2L1D1L3U2D2R2U1D3F3D3 \
Rot2(DL_D,DL_L); \
Rot2(DF_D,DF_F); \
M(F);M(D);M(u);M(R);M(R);M(D);M(D);M(U);M(U);M(L); \
M(D);M(l);M(U);M(U);M(D);M(D);M(R);M(R);M(U);M(d); \
M(f);M(d);
#define F1D1U3R2D2U2L1D2L3U2D2R2U1D3F3D2 \
Rot2(DF_D,DF_F); \
Rot2(DB_D,DB_B); \
M(F);M(D);M(u);M(R);M(R);M(D);M(D);M(U);M(U);M(L); \
M(D);M(D);M(l);M(U);M(U);M(D);M(D);M(R);M(R);M(U); \
M(d);M(f);M(D);M(D);
...much more of the same...
```

[You can find the full source to Bob's solution in our usual online sites. Please see page 2 for details - Ed stb]



To receive information on any products advertised in this issue, send your request via Internet:
productinfo@xplain.com

This monthly column, written by Symantec's Technical Support Engineers, aims to provide you with technical information based on the use of Symantec products.

Q. Why do I get bus errors when I create a CStyleText object by using the constructor with arguments?

A. In the process of upgrading the TCL, constructors with arguments were added to the classes. In this case, constructors with arguments do not create a new handle to a TRec, the macTE data member of the CStyleText object. To work around this, call the constructor with no arguments and call the IStyleTextXO method.

Q. I am having trouble using sizeof() with printf(). For example:

```
printf("char size is %d.", sizeof(char));
```

outputs "char size is 0." Why do I get the wrong result?

A. The output is wrong because the return value of the sizeof() function is a size_t (an unsigned long). Use an %ld rather than a %d as a format specifier. Thus, the correct syntax is

```
"printf("The size of a char is %ld.",
sizeof(char));"
```

Q. How can I avoid problems deleting heap objects whose references are on the stack when using exception handling? The pointer (which is on the stack) to the object becomes invalid when the stack unwinds and only the destructors for automatic objects are guaranteed to be called when an exception is thrown.

A. To handle this, declare pointers as volatile. Last month we explained why you should use the volatile type. Below is a practical example. Use volatile file pointers so that stack unwinding does not reset the value of the pointer (prohibiting the file from being closed).

```
funClass * volatile funClassPtr; // Syntax for volatile declaration
funClassPtr = NULL; // Pointer to NULL guarantees delete as safe.
int myInt = 1; // Watch in debugger to see // the stack unwind.

try_ {
    funClassPtr = TCL_NEW(funClass, ()); //macro for new operator
    myInt = 2; //put new value on stack
    Failure (2, 100); // Force exception
}
catch_all_() { //catch block

delete funClassPtr; //this calls destructor.

// myInt is reset to 1, delete removes the object from the heap
}
end_try_ //end of try block
```

Q. How can I use exception handling without using the Think Class Library?

A. To use exception handling without the Think Class Library, include BRLib and Exceptions.cp in your project. Also, compile with the directive #define NO_TCL

The four macros used to make exception handling work correctly are:

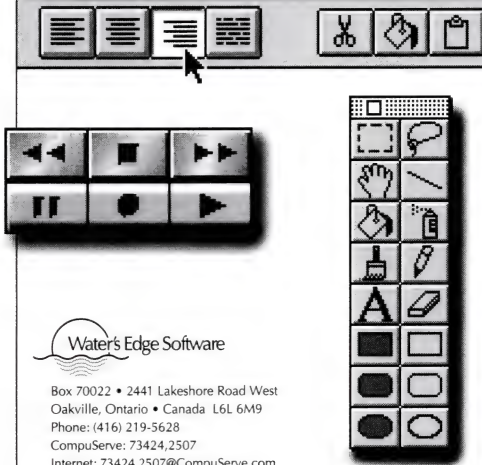
```
AUTO_DESTRUCT_OBJECT
TCL_NEW
TCL_END_CONSTRUCTOR
TCL_START_DESTRUCTOR
```

The macro AUTO_DESTRUCT_OBJECT will guarantee that the destructor is called for an automatic object on the stack. A destructor will only work on a completely constructed object. TCL_END_CONSTRUCTOR helps the compiler to determine the complete construction of an object.

Here's an example that shows how the macros are used.

```
class funClass TCL_AUTO_DESTRUCT_OBJECT //macro in class header
{
public:
    funClass() { // no arg constructor
        cout<< "In constructor."<<endl;
        char * myStr = new char[64]; // allocate memory
        TCL_END_CONSTRUCTOR // End of the constructor
    }
    virtual ~funClass() { // virtual destructor
        TCL_START_DESTRUCTOR // Beginning of the destructor
        cout<< "In destructor."<<endl;
        delete [] myStr; // deallocate memory
    }
};
```

Simply the best GUI Building/Event Managing libraries



Water's Edge Software
Box 70022 • 2441 Lakeshore Road West
Oakville, Ontario • Canada L6L 6M9
Phone: (416) 219-5628
CompuServe: 73424.2507
Internet: 73424.2507@CompuServe.com

Tools Plus for C/C++ or Pascal only \$149 US or \$199 US for both.
(We accept VISA and American Express. Add \$10 for shipping.)

Tools Plus™ 2.5

Tools Plus gives you the routines you need to create a professional looking user interface. Then we make it work. It's that simple.

- For THINK C and Symantec C/C++ (5.0.4 and later) or THINK Pascal
- Over 170 high-powered "set and forget" routines that automate and enhance: event handling, windows, the tool bar, floating palettes, cursors, buttons, picture buttons, scroll bars, menus (pull-down, hierarchical and pop-up), list boxes, fields, Edit menu, clipboard, Dynamic Alerts, and more...
- Easy to learn and easy to use
- Substantial code reduction
- Dramatic code simplification
- Significantly less debugging
- System 6 and 7 compatible
- For novice, intermediate and advanced programmers
- Runs fast; needs little memory or disk space
- Safer than toolbox routines
- No royalties

OK

Language:

- ☐ C/C++
☐ Pascal
☒ Both

Size: 9
10
12
14
18
24
36

- ☒ Saves Time
☒ Saves Money
☒ Easy to Use

Free Evaluation Kit:

CompuServe GO MACDEV,
C & Pascal library, file name: TP252.SEA
AppleLink Software Sampler/Software
Collection/Programmer Tools/Tools Plus
Disk also available by mail.

Q. Where can I get the Quickdraw GX headers?

A. APDA has a Quickdraw GX Developer's Kit which contains a CD with the electronic versions of Inside Macintosh for GX, headers, required system software components, and excellent examples which are compilable under THINK C or Symantec C++. For ordering information, call (800) 282-2732. *If all you need is the header files, check out the MacTech Magazine online sites. See page 2 for details - Ed stb*

Q. When I build an application with the TCL, the application does not seem to be scriptable. How can I make it scriptable?

A. The default flags for the SIZE Resource of the application are set to not receive Background NULL Events. Make sure that the flags for your application have this bit turned on.

Q. I have a program that I'm converting from DOS and would like to be able to draw some simple graphics to the console window. How can I do that?

A. Don't do it. If you draw to the console window, you will not receive update events. However, if you just can't help yourself from going down this path, here is how to do it.

```
#include <iostream.h>

WindowPtr myWindow;           // To be used for the console window.

void main (void) {
    cout << " ";               //A simple way to show the console.
    myWindow = FrontWindow();  //Get a pointer to the console.
    SetPort(myWindow);         //Set the port for drawing.
    PenNormal();               //Set the pen for drawing.
    LineTo(100,47);           //Draw a line.
}
```

Q. If I am mixing C and C++ code, should I turn on the option to use Native Floating Point format?

A. Yes. Symantec's C++ compiler is using the Native Floating Point format. To make sure that your floating point calculations give you the expected results, turn on Use Native Floating Point in the options for the C compiler.

Q. Will Symantec be updating the THINK Reference Databases?

A. Yes. THINK Reference is undergoing a massive overhaul. It will be updated for the Universal Headers and the Think Class Library, and will include new databases for C++ Error Messages. As before, references will be hyperlinked for easy access to information.

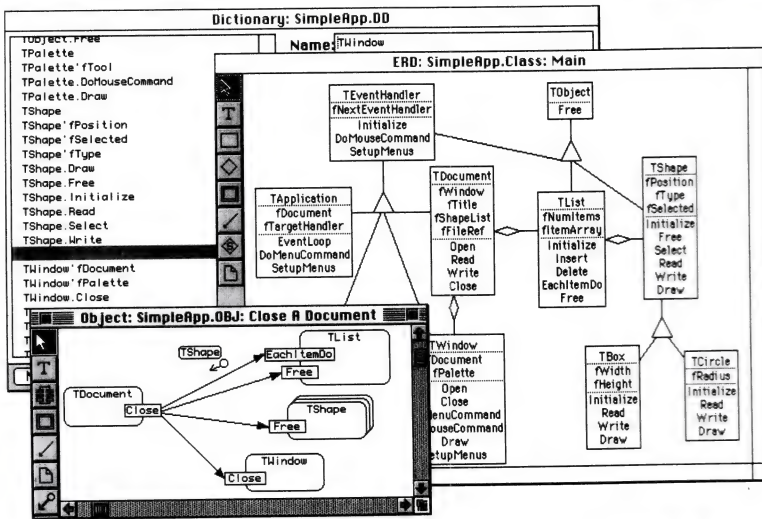
Special thanks to: Craig Conner, Colen Garoutte-Carson, Rick Hartmann, Michael Hopkins, Scott Morison, Celso Barriga, Kevin Irlen, Yuen Li, and Chris Prinos.



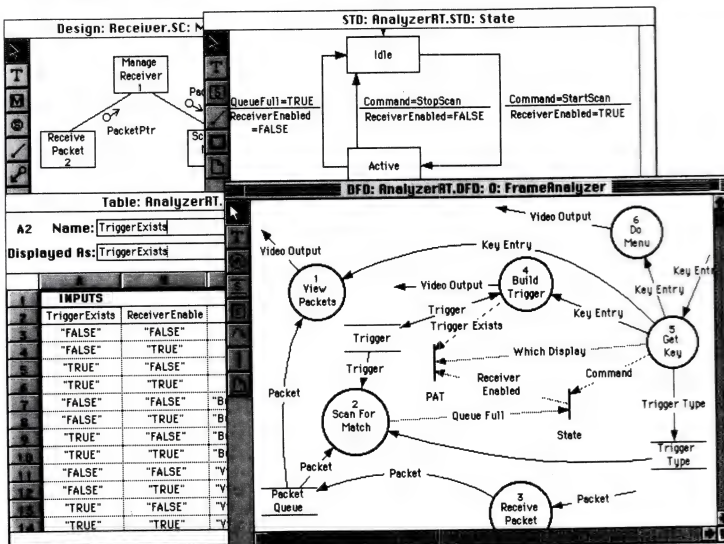
To receive information
on any products
advertised in this issue,
send your request
via Internet:
productinfo@xplain.com

MacAnalyst and MacDesigner

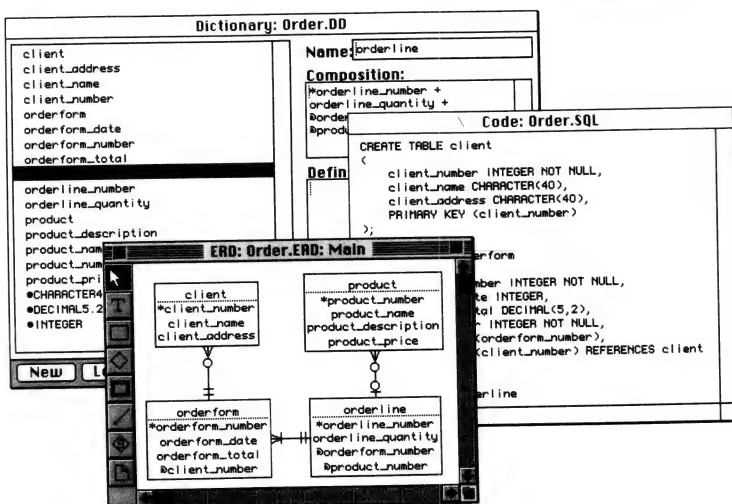
Computer Aided Software Engineering For The Macintosh



OOA/OOD includes OMT, Booch, Coad/Yourdon, Shlaer/Mellor...



Structured A&D using Yourdon/DeMarco, Gane & Sarson, Hatley Pirbhai...



Data modeling and SQL generation for Information Engineering, Chen...

Draw your design, generate code frames, or reengineer existing code back to design diagrams using an integrated tool that supports today's most popular methods.

Software Engineering

Structured Analysis & Design
Object-Oriented Analysis & Design
Real-Time & Task Design
Data & Information Modeling
Screen Prototyping
Integrated Code Editing & Browsing
Multi-User Dictionary & Requirements
Code to Design for C++, Pascal, Fortran...
Design to Code for C++, Pascal, Fortran...

Product Options

MacAnalyst	\$ 995
MacAnalyst/Expert	\$1595
MacDesigner	\$ 995
MacDesigner/Expert	\$1595
MacA&D	\$2995
Translator	\$ 495

Each product is available by single, site, or educational license and supported with on-site training, update service, free newsletter, and technical support.

System Requirements

Macintosh System 7 or A/UX 3 or later and 6 Meg RAM. Also runs on Unix machines with Apple's MAE software.



Winner of 1994 CIO Magazine Readers' Choice Award. Find out why thousands of developers use MacAnalyst and MacDesigner tools for personal computer, mainframe, and embedded software projects. Call today for free technical brochures!

Excel Software

515-752-5359

P.O. Box 1414 • Marshalltown, IA 50158
CASETOOLS@AOL.COM • Fax: 515-752-2435

MacAnalyst, MacDesigner, MacA&D, and Translator are trademarks of Excel Software. All rights reserved.



By Scott T Boyd and John Kawakami

If there's something you'd like to see here, please drop us a note at editorial@xplain.com.

In case you're not familiar with Universal Resource Locator (URL) format, it's essentially

`<servicekind>://<servername>/<pathname>`.

Get yourself a good ftp client (we recommend Anarchie, available at all of your favorite info-mac archives), and a reasonably-priced web browser (we recommend Netscape) and start rummaging around the net. We like to think of it as Mother Nature's hard drive.

Interesting Developer Places

The place to start! Robert Lentz has poured a ton of great information into this web site:

<http://www.astro.nwu.edu/lentz/mac/programming/home-prog.html>

Alpha	ftp://cs.rice.edu/public/Alpha
alt.sources.mac	ftp://ftpbio.bgsu.edu/alt.sources.mac
Apple	http://www.apple.com
see also	http://www.austin.apple.com
see also	http://www.info.apple.com/dev/
Applescript	ftp://gaea.kgs.ukans.edu/applescript
BBEdit	ftp://ftp.netcom.com/pub/bb/bbsw
CodeWarrior	http://www.iquest.com/~fairgate
Celestin's Internet Resources for Mac Developers	http://www.teleport.com/~cci/directories/irfmd/irfmd.html
Dylan	ftp://cambridge.apple.com/pub/dylan
see also	http://legend.gwydion.cs.cmu.edu:8001/dylan
Lisp	http://www.cs.rochester.edu/u/miller/alu.html
MacTechMagazine	ftp://ftp.netcom.com/pub/xp/xplain
Nick's Place	http://www.pitt.edu/~nick/
OpenDoc	ftp://cil.org
MacGL	ftp://ftp.netcom.com/pub/lo/loceff
MacNosy	ftp://ftp.netcom.com/pub/ma/macnosy
Smalltalk	http://www.qks.com
Symantec	ftp://devtools.symantec.com/Macintosh/Updaters/DevTools
TCL stuff	ftp://ics.uci.edu/mac
see also	ftp://daemon.ncsa.uiuc.edu/TCL

Sources

NewsWatcher <ftp://ftp.acns.nwu.edu/pub/newswatcher/>

Info-Mac

info-mac archives <ftp://amug.org/info-mac>
<ftp://mac.archive.umich.edu/mac>

A comprehensive list of Info-Mac and other archives is included with Anarchie, available at these sites or the Peter Lewis site.

We've Moved!

URL's change from time to time and Netcom made sure that ours did. Someday soon, we'll have a home page on a machine of our own, and we'll keep the most current references there. Of course, that means that we'll have to change one more time. In the meantime, you can find us at:

<ftp://ftp.netcom.com/pub/xp/xplain>

People & Places

Apple	ftp://ftp.apple.com/
Apple developer	http://www.support.apple.com
Best of the Net	http://nearnet.gnn.com/gnn/gnn.html
Bill Modesitt	ftp://ftp.maui.com/pub/maui-sw
Consensus	http://www.consensus.com:8300
Paul Robichaux	http://www.iquest.com/~fairgate
Peter Lewis tcp/ip apps	ftp://amug.org/pub/peterlewis
QuickCam	http://www.engin.umich.edu/~friscor/QuickCamtm/readme.html

Macintosh General Stuff

Macintosh Vendor Directory
<http://rever.nmsu.edu/~elharo/faq/vendor.html>

Newsgroups

comp.sys.mac.programmer.*, where * is "digests", "info", "help", "tools", or "misc", are the main Macintosh programmer hangouts.

Getting a nicer Internet interface

The Internet Adapter turns a simple unix shell account into a SLIP connection. Telnet to tia.marketplace.com, or point your www browser to marketplace.com for info. *I've been using it for a month and have had no problems. Consider this a thumbs up from a satisfied customer - jkl*

ISDN = Faster Internet on the cheap

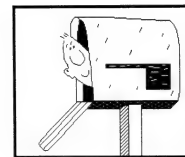
The next hot topic in home and home office Internet looks like ISDN. You just knew the phone company would finally figure out what to do with this technology! ISDN prices are dropping fast. For an incredibly comprehensive index of ISDN info, check out <http://alumni.caltech.edu/~dank/isdn/>

Don't try this at home

If this is any indication, the phonebook of the future is going to have a lot more than a first initial, last name, and a phone number: <http://www.umich.edu/~dugsong/index.html#ButterBoy>



By Scott T Boyd, Editor



EVENBETTERBUSERROR AGAIN?

Your printed copy of EvenBetterBusError in the December issue has a bug. Your constant `SizeOfCodeBlock` is computed to include only the code, but then is used as if it contained both the code and the constant `BuggyCodeWroteToNil`. The effective address of the PEA just before `_DebugStr` in the VBL task is outside the system heap block you obtained with `_NewPtr`.

Check it out. The effect is that after legitimately dropping into the debugger, one might see almost anything as the debugging string, hardly helpful!

Fix it by moving the `EndOfCodeBlock` label after the DC, not before. Does your assembler complain about labels with no contents? Fix that by putting another dummy DC after `EndOfCodeBlock`.

I wish we could ignore the dumbbells who claim their bugs are caused by the detection mechanism, but what if one needs their software?!

– Emerson Mitchell, 72257.2213@compuserve.com

[Yes, indeed, it's a bug. It's also missing an ALIGN to get it on a nice boundary. That'll teach me to assemble and link without actually installing and running. Another lesson learned. Thanks for the correction. As for the dumbbells, we can't fix them, but a whole lot of folks now know to use EBBE when testing software (theirs and others). The "culprit" that I wrote about has already rewritten chunks of their product and has a new appreciation for low-level debugging tools. If you see other software that could use fixes along these lines, please drop us a note – Ed stb]

THIS IS CALIFORNIA, AND IT'S 1994

Scott, 14.4K Internet access? Why bother, when PacBell will install an ISDN line in your home for less than \$100, and companies like Internex will give you ISDN access for \$49/month?

That's what I did. With an Ascend router, I get ISDN into my local Ethernet, and we both have high speed access. Whee!

– David Ramsey, ramsey@be.com

[That sounds terrific, but I want to host my own Internet services (Web server, ftp site, and so forth). I can do those now, albeit slowly, with my constant 14.4K connection. A constant ISDN connection in this area would run more than three times what I'm paying now. ISDN equipment costs substantially more than my cheap modem, too. Ascend routers get you up and over the \$1000 mark pretty quickly. Of course, these reasons don't mean I'm not jealous! – Ed stb]

ANOTHER BUG?

I just discovered a nasty "gotcha" in a tip I submitted to MacTech, which was printed a few months ago.

If you hack a folder alias with ResEdit to look as if it were an alias to the Desktop Folder, don't ever ask Finder to "Get Info..." on the hacked alias and then try to "Find Original." If you do, Finder crashes; dramatically or quietly, depending on its mood at the time.

I hope this doesn't mean I have to give back the "tip of the month" money for that one, 'cuz I already spent it!

– Lee David Rimar

FTP'S A GOOD IDEA, BUT...

I'd like to thank you for maintaining an ftp-site, and a net-presence in general. I would, however, encourage you to seek a mirror site. I've had very little luck connecting to ftp.netcom.com. And most of the few times I have been able to connect, the response time has been unbearably slow.

– Robert Fisher, malirath@zilker.net

[We don't know what's up with Netcom, but we're not happy with it either. We'll be getting our own site online sometime soon. Thanks for the suggestion about a mirror site – we'll keep you posted – Ed stb]

DEVELOP AND MACTECH EDITORS READ EACH OTHER

In response to Steve Kiene's letter in the December issue:

I enjoyed Steve Kiene's well-written letter to MacTech in the December issue – even though I'm the editor of Apple's own technical journal, **develop**. We try not to make **develop** a Stars and Stripes, feel-good magazine either, although we do need to push the company's technology direction a bit to avoid incompatibilities for developers down the road.

My forthcoming editorial in Issue 21 of **develop** is on the subject of my own emotional attachment to the Mac, so I especially enjoyed the paragraph on your similar attachment. I hope your attachment remains – and that you're a **develop** reader :-)

Regards,

– Caroline Rose, CROSE@applelink.apple.com

LIST VERSUS OBJECT IN SMALLTALK

Kevin O'Neill wrote:

I managed to get the time to read your article in MacTech last night. It was great and I hope to see more.

I'd like to know more about how you decided to use a list to store the drag data rather than an object of a specific class. I found myself flipping back to the stop where you defined the list so that I could be sure what was being referred to in the various 'data@x' statements.

Actually it turns out that we should have used a subclass of `<List>` which we call a `<DragPackage>` class. At the time it wasn't clear whether it would be a private data structure for use among a few objects, or whether it would have a public/client API. Since the time that article was written we have concluded that there really should be some behavior attached to the drag-data to enable it to serve a more general usage.

:-) That was good instinct/observation on your part...

Thanks for the comments and feedback.

– Dave S., quasar@qks.com

EVENBETTERBUSERROR YET AGAIN (EBBEYA)

The wrath over EvenBetterBusError (Editor's Page, November 1994) is directed to the wrong target. The typical Mac developer, struggling against all odds to earn a living from a platform intentionally restrained

DragInstall makes your installer choice easy



Easy for your users

DragInstall's unique drag-and-drop interface makes installing your software a piece of cake for your users. And cuts down on your tech support calls.

Easy for you

DragInstall's Builder utility allows you to create complex installers in minutes not days. And without the need to learn a complicated scripting language.

Easy on your wallet

DragInstall's one-time fee of only \$300.00 allows you to distribute an unlimited number of installers. No yearly renewals, no royalties, no hassle!

To find out just how easy DragInstall is, call us at
1-800-890-9880
to receive a free demo disk.
Or contact us at:

Ray Sauers Associates, Inc.
1187 Main Avenue, Suite 1B
Clifton, NJ 07011-2252 USA
Voice: 201-478-1970
Fax: 201-478-1513
AppleLink: D1922
Compuserve: 70731,2326
Internet: sauers@aol.com

to single-digit shares of the business market, is not helped by the thought police yelling "bonehead" and "liar" at him. Anyone who says of a block of 68K code, "As you can see, it's pretty simple" is well out of touch with the real world of business applications programming.

Rather than excoriate the poor developer, why not direct your flame towards the source of the problem, the tool developers? EvenBetterBusError is just one of a scattered set of poorly-documented, poorly-designed and poorly-distributed hacks, each of which is intended to address some gaping problem in Mac development tools. Why can't the functions of all such debugging aids be bundled with the tools themselves, in a simple-to-use format? Why don't more development environments automatically (or optionally) check for such detectable "nefarious acts" as using DisposeHandle on a resource, dereferencing NIL, or calling DisposeHandle twice? Why isn't a simple, high-level form of Discipline standard with all languages? Finally, why do toolmakers treat source-level debugging as unmanly, and therefore of no interest? Some of these hacks give results in a machine level so low as to be useful to only a smattering of programmers - I guess that leaves the rest of us as "boneheads".

Think Pascal was (as is) fairly good on some of these issues, so why have more recent environments taken steps backwards from that level of assistance? It should not be necessary for the beleaguered Mac developer to conduct a fishing expedition through CDs and online sources to find all of the hacks needed to fix the obvious omissions of the development environments.

- Kevin Killion Stone House Systems, Inc. sbs@mcs.com

[Thanks for your excellent letter. A few thoughts came to mind while I was reading your letter, and here they are - Ed stb]

First, someone *has* (as you may have already noticed in the December issue) put a bunch of the tools together in the form of QC™. They brought together many of the tools, wrapped them together in a nice package, and are doing what they can to get them out to developers. At \$100, it's cheap at twice the price.

Second, I'll cop to playing the role of thought police. Somebody has to do it. Apple sure isn't, and, as you pointed out, neither is Symantec or Metrowerks. The aspersions were cast at a vendor who chose to blame the tool rather than address their serious, yet known bug. Ignorance is one thing, but blaming EBBE was a wrong thing to do.

In a followup conversation with that particular vendor, the

product manager told me that he was "horrified" once he had read my editorial. The right thing has happened now - the product has been repaired, and we're a little better off for them better understanding the situation, the available tools, and their responsibility to test their software with whatever tools are available.

Third, let's not blame someone for the existing tools being poorly documented and poorly distributed. Why? Greg, Bo3b, and others kindly and generously donated tools. Those tools are available on every online service and on Apple's CDs, as well as our online sites. Sure, their documentation isn't great, but at least it exists to some degree. I'm sure you didn't mean to put them down, but, having been there when these tools were written (and having written the beginnings of one of them myself), let's praise them for not only making us aware of the problems, but also giving us tools (albeit crude) to address the problems.

Third-and-a-half, the three principal problems that DoubleTrouble, DisposeResource, and EBBE address are problems that we (the System 7 team at the time) realized were bigger problems than we had known during the System 7 effort. Greg wrote the tools after he had debugged hundreds of "incompatible" pieces of software. Sometimes it just takes a while to realize the need for a specific tool. Without his debugging efforts, we still wouldn't have these tools.

Fourth, I totally agree that environment vendors need to provide more and better tools. Some vendors have. For example, QKS SmalltalkAgents knows about various classes of memory, allocates most things for you, and deallocates when your done. It's nearly impossible to make some of these classic mistakes with an environment like STA.

Finally, I'd say that it's time to put some blame on Apple. Isn't it time that they develop a system which makes it nearly impossible to make these kinds of mistakes? Now that we've had a decade's worth of experience, don't we have a pretty good idea what the common mistakes and failure modes are? Couldn't we find a way to eliminate some of them? QKS has. Dylan does. NeXT has. Sun is working on it. Taligent has put a lot of effort into it. When is Apple going to step out and take a leadership role in making the developer's job easier, safer, and more rewarding?

- Ed stb





By Scott T Boyd, Editor

URL STANDARD APPLE EVENT SUITE

Uniform Resource Locators (URLs) are a standard notation for identifying the locations of files and other resources on the Internet. Mac TCP/IP networking programs often make use of each other as "helper programs". To make it easier to do this, some Mac Internet software developers have designed a simple standard for Apple Events programs can send to each other to ask them to process URLs. Here's a brief overview of the events. Suite code: 'GURL'

The geturl event

Get an object referenced by an URL and display it in a window or save it to a file:

```
geturl <URL:...> [to <file>]
```

retrieve the object reference by the URL

Result: small integer – result code

The fetchurl event

Get an object referenced by an URL and return the object as the event result

```
fetchurl <URL:...>
```

Result: the referenced object, usually text

Servers must support the following formats for the URLs:

- (1) scheme:... (the "canonical form")
- (2) <scheme:...>
- (3) URL:scheme:...
- (4) <URL:scheme:...>

This standard was designed by John Hardin, Peter Lewis, Steve Dorner, Farhad Anklesarian, Aleksandar Totic, and other Mac TCP/IP developers. It was edited by John Norstad <URL:mailto:j-norstad@nwu.edu>, and is available in full at:

ftp://ftp.acns.nwu.edu/pub/newswatcher/url-ae-standard.txt

BBEDIT 3.1

Bare Bones Software, Inc. announced version 3.1 of their *BBEdit* text editor. The new version includes "soft" text wrapping, interaction with development environments not previously supported, electronic documentation, and an all-new packaging and delivery system.

BBEdit 3.1 features new "soft" text wrapping. Previous versions of BBEdit required users to insert carriage returns in order to break lines, and to manually re-format the text while editing. BBEdit 3.1 now offers the option to wrap lines without inserting carriage returns, and will automatically re-flow lines as necessary when the user inserts or deletes text.

BBEdit now supports integration with THINK C, Symantec C++, and CodeWarrior. BBEdit provides a unified user interface for interacting with all supported environments, and lets the user easily switch between them. Metrowerks

added services in the new CW5 CodeWarrior release to support close integration. BBEdit 3.1 also supports the upcoming version 8.0 of Symantec C++ for Power Macintosh.

The new version of BBEdit ships on CD-ROM, and includes full documentation for BBEdit in machine-readable form (a printed and bound manual is available at a nominal extra cost), demos of products from Bare Bones Software and other developers, promotional information and special offers from various third-parties, and a collection of BBEdit extensions contributed by BBEdit users from all over the world.

SRP US\$119. Customers who purchase BBEdit 3.0 (the current version) after December 1, 1994 will be eligible to receive a free upgrade to version 3.1. All other owners of BBEdit 3.0 will be able to upgrade to the new version for US\$39. An upgrade path is also available for users of older versions of BBEdit (including freeware versions), and for selected competing and complementary products; contact Bare Bones Software for more information.

Bare Bones Software, Inc. P.O. Box 108 Bedford, MA 01730 (508) 651-3561 voice, (508) 651-7584 fax. E-mail bbsw@netcom.com, AppleLink BARE.BONES, eWorld BareBones, CIS: 73051,3255.

THE INTERNET CONFIGURATION SYSTEM

Quinn "The Eskimo" announces the release of the Internet Config system, a development that makes Internet access by a Macintosh even easier.

We all use many different programs to access the Internet and each of these programs has its own preference dialog, wherein you set things like your Email address, your FTP helper application and your preferred program to open ".jpg" files. Keeping these preferences in sync in all your Internet applications is increasingly difficult. Worse yet, many simple applications do not even have a mechanism for setting these preferences and so you are stuck with the author's default preferences.

The Internet Configuration system is a solution to this problem. Internet Config is an application that allows you to set these preferences once. Internet Config stores these preferences in a shared database and any "IC-Aware" application will get its preferences from this database.

Internet Config has broad-based support from a wide range of Macintosh Internet developers. A number of applications have already been programmed to be "IC-Aware" and many more are expected soon.

Internet Config will run on all Macintosh Plus or newer machines running System 6 or later. Internet Config is available at the MacGifts, Info-Mac and UMich ftp sites (and their mirror sites), and also any site that holds Peter Lewis' software (see

p. 66). Internet Config is available as a NewsWatcher helper at <ftp://ftp.acns.nwu.edu/pub/newswatcher/helpers/> and on the TidBITS site <ftp://ftp.tidbits.com/pub/tidbits/>

Internet Config has been placed in the public domain and can be freely redistributed by any means. This means that you're legally entitled to:

- sell it or its source code commercially
- distribute it as part of any other product
- distribute it on CD, disk, network or any other medium
- do anything else you like with it

From a programmer's point of view, Internet Config is an application programmer interface (API) that lets you read and write shared preferences. This API calls through to the Internet Config component (if it is present) so that the actual implementation of the preference database code is dynamically linked in to your application. This means that as the Internet Config system gets smarter, your applications will become smarter, without the need for any work on your part (other than supporting Internet Config in the first place).

The Internet Config Application automatically installs the Internet Config component when it is first run. If the Internet Config component is not available, then the API will use a statically linked version of the current database code, so IC-Aware programs do not rely on having the component available.

Internet Config manages the following groups of preferences: Personal, Email, News, File Transfer, Other Services (e.g. Gopher and Ph), Fonts, File Types, and Helpers (for mapping URLs to their help applications).

Full source code to the Internet Config system has been placed in the public domain. The system is essentially 'open', not a proprietary add-in which may become a liability in the future.

You can get the Internet Config Programmer's Kit from <ftp://ftp.share.com/internet-configuration/> and <ftp://redback.cs.uwa.edu.au/Others/Quinn/Config/>

It has all the information you need to develop for Internet Config in Pascal or C using any of the common development environments (Metrowerks, Think and MPW). The kit also contains the source code to the Internet Config Extension. The same sites also have the Internet Config Application Source Kit, which contains the source code to the Internet Config application (in Think Pascal).

About adding IC support to NewsWatcher, "I'm pleased, too. I figured this would be reasonably easy to support, and it turned out to be even easier. There were no major problems or stumbling blocks – just a bunch of really easy code, and it worked with no major hassles." – John Norstad

HIERARCHICAL OR RELATIONAL? – WHY NOT HAVE BOTH?

iD de Magellan is a new external package for 4th Dimension. It allows 4D programmers to manage and present their relational data in a hierarchical manner. Whatever the structure of the data, iD de Magellan can create and maintain a hierarchical representation of it. Relational and hierarchical

database systems, the two most common types in today's database market, are normally completely separate. iD de Magellan allows you to integrate the two representations.

iD de Magellan adapts itself to any relational structure, generating a hierarchical representation of the data. Moreover, iD de Magellan can generate more than one hierarchical representation of the same relational structure. Once integrated into an external zone, iD de Magellan manages all data processing, navigation, formatting, importing, exporting, printing, etc., in an outliner-like interface.

Logiciels Magellan (514) 344-1056 voice, (514) 344-2970 fax, AppleLink LOG.MAGELLAN, CompuServe 76506,1656.

Demo available (\$10 CDN) by post to 919 Dunlop ave. Outremont, QC Canada H2V 2W9, or on CompuServe/MACDEV/Library 6, file name: ID.SEA, or AppleLink/Third Parties/ACI/ACI Information/Third-Party Information.

APPLE SUPPORT PROGRAMS UPDATE

Apple Computer introduced the **Newton Associates Program**, and enhancements to the Newton Partners and Macintosh Partners programs. The Newton Associates Program is a low cost (US\$400 annual fee), high quality, self-help development support program for Newton developers, and includes:

- Support services from Apple's Developer Support Center
- Discounted rates for online technical information
- Access to a technical Q&A reference library
- Discounts on Newton and Macintosh hardware
- A Newton Orientation Kit
- A monthly Newton developer mailing, including the Newton Developer CD and *Newton Technology Journal*
- Use of Apple's third party compatibility lab
- Discount on a Newton development class
- Invitation to Newton and Worldwide Developer Conferences
- Eligibility to participate in StarCore's Affiliate Label Program

The **Newton Partners Program** (formerly PIE Partners Program) price is now US\$2500 annually, and includes all the features of the Newton Associates Program plus services such as expert-level programming support via e-mail, free updates to Newton development tools, and participation in select Apple marketing and PR opportunities.

The **Macintosh Partners Program** has been enhanced with new features such as seeding on most Macintosh technologies, pre-release documentation for new CPU's, one free Mac OS SDK subscription. Macintosh Associates may purchase the Mac OS SDK subscription at a substantial discount directly from APDA.

The **Apple Multimedia Program** (AMP) no longer requires membership in the Associates Program. Members of the Apple Multimedia Program will continue to receive all the core services available through the Apple Developer Programs, as well as the benefits received through the Apple Multimedia Program.

For more information on joining the Newton Associates Program or any of the Apple Developer Programs, please contact the Apple Developer Support Center at (408)974-4897, link DEVSUPPORT or e-mail devsupport@applelink.apple.com.



MacRegistry™ Developer Job Opportunities



If you are a Macintosh developer, you should register with us! We have a database that enables us to let you know about job opportunities. When we are asked to do a search by a client company the database is the first place we go. There is no charge for registering. The database service is free. Geographic Coverage is nationwide.

Marketability Assessment - To get a specific feel for your marketability send a resumé via Email or call. You may also request a Resume Workbook & Career Planner.

Discreet - We are very careful to protect the confidentiality of a currently employed developer.

Scientific Placement is managed by graduate engineers, we enjoy a reputation for competent & professional job placement services and we are Mac fanatics.

1-800-231-5920 | AppleLink: D1580 | 713-496-0373 fax

Scientific Placement, Inc.

Dept. MT MacRegistry
P.O. Box 19949
Houston, Texas 77224
(713) 496-6100
das@scientific.com

Dept. MT MacRegistry
P.O. Box 71
San Ramon, CA 94583
510-733-6168
bge@scientific.com

Dept. MT MacRegistry
P.O. Box 4270
Johnson City, TN 37602
(615) 854-9444
rjg@scientific.com

MAC PROFESSIONALS

Manpower Technical, a leader in the Technical Services industry, has current and upcoming contract openings for experienced MAC personnel with the following expertise: Software Development, Network Administrators, Desk Top Publishing, Help Desk, Applications Support, Technical Writers/Editors.

Positions are in California and throughout North America.

Interested persons are encouraged to send their resume to:

Manpower Technical
ATTN: Sandra Anderson
P.O. Box 2053
Milwaukee, WI 53201

(800) 558-6992
Fax # (414) 332-0378

Take Off!

Russ LaValle,
MacApp
Programmer at
MacXperts, and
F-14 fighter pilot
in Desert Storm

"I've had SAMS
shoot at me, but
coding is more
exciting than
tweaking a solid
piece of code"

MacApp Programmers Wanted

MacXperts has openings for experienced C++ and MacApp programmers. If you have what it takes, and the desire to achieve, call Kendall Tyler at MacXperts.

Voice: 800-358-8040 Fax: 804-358-3847
Internet: xperts@inf.net
AppleLink: xperts AOL: MacXperts

Get Your Name "In Lights"

Have you ever thought about writing an article? The editorial staff at MacTech Magazine would like to personally invite you to write for the publication. While writing an article is not difficult, it does take a bit of time. But, you get to share your knowledge with the community, see your names "in lights", and ... you get paid for your efforts!

To get started, you can download the MacTech Writer's Kit from one of our online support areas. Here you will find information on how to submit an article - and it comes with examples, templates and style sheets. Feel free to e-mail us with questions.

Call for Articles

Many of you have asked "what topics would we like to see?" Recently, our Editor, Scott T Boyd, provided us with insight as to what he'd like the magazine to cover in 1995. Our goal at the magazine is to publish articles on diverse topics that are sure to keep our readers both interested and well-informed throughout the year. If you'd like to be part of this group, think about what you know about - that's what we'll be most interested in. As a guideline, you can think about one of the following issues.

For example, topics that include articles that teach debugging techniques and advocate good debugging tools. This continues a recent theme about making software more reliable. We will continue to talk about "cool" Apple technology such as Threads and Drag and Drop; or as Scott said, "software that takes us out of the Stone Age."

There will be tips on how to get your business on the Internet and how business models are shaping up. More heavy-hitting deep technology articles will be included, such as the recent September article on emulator technology and the two-part article on PowerPC Architecture. We'll see continued coverage on the developments in OpenDoc and OLE technologies, as those on the sidelines start choosing up sides or deciding not to play.

MacTCP is another area of importance, as more people get excited about providing Macintosh-quality software for Internet users. You will see additional coverage on Visual Programming. You should also expect to see more on cross platform development. And, even though it's a moving target, we will be covering Apple's next System Software release - Copland.

E-mail us with your idea and let's talk. You never know, you might get your name "in lights"!

If you have a CD-ROM drive, then you've gotta have every article published in the first 9 years of MacTech Magazine!!

**FREE
UPGRADE TO
VOL 1-10!***

... now in THINK Reference format!

**Available
Now!!**

MacTech CD-ROM, Volumes 1-9:

**Over 1100 Articles.
All 103 Issues,
including all of 1993.
All the Source Code.
THINK Reference 2.0.
Working Applications.
Full Documentation.
Demos for Developers.
And More!!**

"When I designed THINK Reference, I envisioned endless databases at my fingertips. MacTech has doubled the information that is just a mouse click away."

— Darrell LeBlanc, Formerly of Symantec,
Author, THINK Reference 2.0

"The CD strikes me as an impressive and very useful resource. The only disadvantage I've found is that each answer the databases yield exposes me to so many more issues, that I find myself exploring the articles for the sheer wonder of it, and thus putting off the real coding I should be doing. :-)"

— Nicholas De Mello
MacTech CD Beta Tester

MacTech Formerly MacTutor **MAGAZINE™**
FOR MACINTOSH PROGRAMMERS & DEVELOPERS

Voice: 310/575-4343 • Fax: 310/575-0925
AppleLink: MT.CUSTSVC
CompuServe: 71333,1063
Internet: custservice@xplain.com
America Online & GENie: MACTECHMAG

✓ **45 MB of MacTech Magazine articles, Volumes 1-9**

Every article, 1100+ of them, from all 103 issues of MacTech Magazine printed from 1984 through 1993. Articles ranging from Assembly to BASIC, C to Pascal, Fort to FORTRAN and more. *And the articles are in THINK Reference!*

✓ **Hyperlinks to Inside Macintosh Databases of THINK Reference**

The articles have hyperlinks to relevant portions of the *Inside Macintosh* databases of THINK Reference. For example, if you are looking for information on Aliases, look at the MacTech articles on Aliases and use the hyperlinks to jump to the *Inside Macintosh* entry for the Alias Manager. And now, with the articles in THINK Reference, you can do free-text searches 8-10 times faster than you could previously with On Location™ 2.0.

✓ **100 MB of MacTech Magazine source code examples, samples, and utilities.**

These are the files that go with the magazine – the code that the articles are talking about. Use them in your own applications, with no royalties!

✓ **A fully-capable version of THINK Reference 2.0.3**

Including *Inside Macintosh* Volumes 1-6 (7 MB).

✓ **Sprocket – MacTech's Tiny Framework.**

Build your simple application using a lean, mean application framework. Experiment with new code without having to build a whole application around it!

✓ **80 MB of FrameWorks, MacApp®, MADA and SFA articles, files and source code.**

The most complete set of FrameWorks archives known.

✓ **Apple APIs, Utilities, and SDKs**

Including: Universal Header Files, Discipline, Macintosh Drag and Drop SDK, MacsBug, Telephone Manager SDK, Thread Manager SDK, TrueEdit 1.8 and and more.

✓ **Ariel Publishing's Inside BASIC on disk**

...and other related BASIC programming information and tools.

✓ **75 MB of Special Demos relevant for developers**

MacTech CD-ROM, Volumes 1-9:

\$199 plus shipping and handling. \$69 plus shipping and handling for upgrades from any previous version of the CD

* All purchasers of the 1-9 CD will receive a FREE upgrade to the next version of the CD when it becomes available.

FREE! THINK™ Reference

Symantec's THINK Reference 2.0. Complete on-line guide to *Inside Macintosh*, Vol. I-VI, with cross referenced index, detailed information of each function, procedure and detail needed when programming the Macintosh.

SYMANTEC.™



MACTECH MAGAZINE PRODUCTS & ORDER INFORMATION

E-mail, Fax, write, or call us. You may use your VISA, MasterCard or American Express; or you may send check or money order (in US funds only): MacTech Magazine, P.O. Box 250055, Los Angeles, CA 90025-9555. Voice: 310/575-4343 • Fax: 310/575-0925

If you are an e-mail user, you can place orders or contact customer service at:

- **AppleLink:** MT.CUSTSVC
- **CompuServe:** 71333,1063
- **Internet:** custservice@xplain.com
- **America Online:** MT CUSTSVC
- **GEnie:** MACTECHMAG

SUBSCRIPTIONS

US Magazine: \$47 for 12 issues
Canada: \$59 for 12 issues
International: \$97 for 12 issues
Domestic source code disk: \$77 for 12 issues
Int'l source code disk: \$97 for 12 issues

CD-ROM

MacTech CD-ROM, Volumes I-IX: Includes over 1100 articles from all 103 issues (1984-1993) of MacTutor Magazine (formerly MacTutor). All article text and source code. Now in THINK Reference format. The CD includes Symantec's THINK™ Reference 2.0, working applications with full documentation, product demos for developers and more. See advertisement, this issue: \$199. Upgrades \$69, e-mail, call or write for info.

BOOKS

The Best of MacTutor, Volume 1: **Sold Out**
The Complete MacTutor, Volume 2: **Sold Out**
The Essential MacTutor, Volume 3: \$19.95
The Definitive MacTutor, Volume 4: \$24.95
The Best of MacTutor, Volume 5: \$34.95
Best of MacTutor Collection, Volumes 3 - 5: \$69
Best of MacTutor, Volumes 6, 7, 8 & 9: Not available

DISKS

Source Code Disks: \$8 each
Topical Index (1984-1991) on disk: \$5

MAGAZINE BACK ISSUES

Volumes 3, 4, 5, 6, 7, 8, 9 and 10: \$5 each (subject to availability)

SHIPPING, HANDLING & TAXES

California:

Source disk or single issue: \$3
Single book or multiple back issues: \$5
Two books: \$8 • All other orders: \$12

California residents include 8.25% sales tax on all software, disks and books.

Continental US:

Source disk or single issue: \$3
Single book or multiple back issues: \$7
Two books: \$15 • All other orders: \$17

Canada, Mexico and Overseas: Please contact us for shipping information.

Allow up to 2 weeks for standard domestic orders, more time for international orders.

PLEASE NOTE

Source code disks and journals from MacTech Magazine are licensed to the purchaser for private use only and are not to be copied for commercial gain. However, the code contained therein may be included, if properly acknowledged, in commercial products at no additional charge. All prices are subject to change without notice.

10% OFF
ALL BOOKS!

3RD PARTY PRODUCTS

MACTECH EXCLUSIVES

MacTech Magazine is your exclusive source for these specific products:

NEW! Ad Lib 2.0 The premier MacApp 3.0 compatible ViewEdit replacement. A powerful user-interface editing tool to build views for MacApp 3.0 and 3.1. Ad Lib allows subclassing of all of MacApp's view classes including adorners, behaviors, and drawing environments. String and text style resources are managed automatically. Alternate display methods, such as a view hierarchy window, allow easy examination of complex view structures. Ad Lib includes source code for MacApp extensions that are supported by the editor - buttons can be activated by keystrokes, behaviors can be attached to the application object, and general purpose behaviors can be configured to perform a number of useful functions. Run mode allows the user to try out the views as they will work in an application. Templates can be created to add additional data fields to view classes. Editing palettes provide fast and easy editing of common objects and attributes. Works with ACI's Object Master (version 2.0 and later) to navigate a project's user interface source code. \$195

NEW! FrameWorks Magazine: \$8/issue, subject to availability.

NEW! FrameWorks Source Code Disk: \$10/issue, subject to availability.

NEW! Five Years of Objects CD-ROM: FrameWorks archives and source code from April 1991 to January 1993, plus selected object-oriented publicly available software and demos. \$95

MADACON '93 CD-ROM: The highlights of MADACON '93, including Mike Potel on Pink, Bedrock, MacApp, OODLs, and more. Slides, articles, demos, audio, and QuickTime. \$95

NEW! MAScript 1.2 adds support for AppleScript to your MacApp 3.0.1 and 3.1 based applications. Make your application scriptable and recordable by building on a tried and tested framework for object model support. MAScript dispatches Apple events to the appropriate objects, creates object specifiers, and makes framework objects like windows and documents scriptable and recordable. Sample

application shows you how to begin adding support for scripting and recording. MAScript includes complete source code. Install MAScript by modifying one MacApp source file, then adding another to your project. Future versions of MacApp will incorporate MAScript, so MAScript support you add now will work in the future. \$199

NEW! The Mjølner BETA System is a software

development environment supporting object-oriented programming in the BETA programming language. BETA is uniquely expressive and orthogonal. BETA unifies just about every abstraction mechanism - including class, procedure, function, coroutine, process and exception - into the ultimate abstraction mechanism: the pattern. BETA includes: general block structure, strong typing, whole/part objects. The compiler: binary code generation, automatic garbage collection, separate compilation, interface to C, Pascal, and assembler.

The system: persistent objects, basic libraries with containers classes, platform-independent GUI application frameworks on Unix, Mac and Windows NT, metaprogramming system. The tools available on Unix: the hyper structure editor supporting syntax directed editing, browsing, etc., and the source code debugger are currently being ported to the Macintosh system. The Mjølner BETA System for Macintosh requires MPW (basic set) 3.2 or later.

Package containing compiler, basic libraries, persistent store, GUI framework, and comprehensive documentation. (Other packages are also available) \$295

NEW Version! Savvy 1.1

OSA support includes attachability, recordability, scriptability, coercion, in addition to script execution, idling and i/o. Apple event support includes complex object specifiers, synchronous/asynchronous Apple event handling, and Apple event transactions for clients and servers. The Core Suite of Apple event objects is supporting including the application, documents, windows, and files. Documentation includes technology overview, cookbook, and sample code. \$250 Savvy now supports MPW 3.1, 3.11 and continues to support 3.01, as well as supporting Metrowerks Code Warrior. **This month only, special offer** - All Savvy versions include free

MAIL ORDER STORE

copy of Savvy QuickTime!

NEW! **More Savvy** includes all Savvy features plus Apple event support for all sub-classes of TEventHandler with extensive view support. Apple event support for text includes text attributes and sub-range specification. Recordability supports additional actions, and coercion includes additional types. Additional client and server Apple events. \$450

NEW! **Super Savvy** includes all More Savvy features plus compile, edit, and record scripts using built in script editor. View template editors, like Ad Lib, can attach scripts to view objects and modified scripts are saved with the document. Script action behavior allow quick access for executing and editing scripts attached to views. Text to object specifier coercion plus more. \$700

NEW! **Savvy QuickTime** Requires Savvy, More Savvy, or Super Savvy. Includes QuickTime, Apple event and view template support. Movies come out of the box ready to play, edit, and react to Apple events. They can be included in any view structure, including templates, and are displayed in the scrap view. Movie controls include volume, play rate, looping mode, display style, and other characteristics. \$250

NEW! **Savvy DataBase** Requires Savvy, More Savvy, or Super Savvy. Available Winter 1994. \$250

MacTech Magazine is your exclusive source for available back issues of SFA's magazine, source code disks and assorted CD's. Call for more info and pricing.

BOOKS

Defying Gravity: The Making of Newton Doug Menuez and Markos Kounalakis. An in depth, dramatic account of the story of Newton's creation. It is a techno-logical adventure story; a fascinating case study of the process by which an idea is born and then translated into a product on which careers and fortunes can be made or lost. It is a new kind of business book, one that captures through powerful photo-journalism and a fast-paced text, the human drama and risk involved in the invention of a new technology for a new marketplace. 196 pgs., ~~\$20.95~~ **\$26.95**

The Elements of E-Mail Style by Brent Heslop and David Angell. Learn the rules of the road in the e-mail age. Concise, easy-to-use format explaining essential e-mail guidelines and rules. It covers style, tone, typography, formatting, politics and etiquette. It also outlines basic rules of composition within the special context of writing e-mail and includes samples and templates for writing specific types of e-mail correspondence. 208 pages. ~~\$14.95~~ **\$13.45**

NEW! **E-Mail Essentials** by Ed Tittel & Margaret Robbins is a hands-on guide to the basics of e-mail, the ubiquitous networks communication system. The book is suitable for both the casual e-mailer and the networking professional, as it covers everything from the installation of e-mail to the maintenance and management of e-mail hubs and message servers. The book explains the fundamental concepts and technologies of electronic mail, featuring chapters on Lotus applications and CompuServe, as well as information on upgrading, automation, message-based applications, and user training. E-mail is a step-by-step, jargon-free guide that will

enable the e-mail user to get the most out of the communication potentials of networking. 250 pp. ~~\$24.95~~ **\$22.45**

NEW! **Graphics Gems IV** edited by Paul Heckbert Volume IV is the newest collection of carefully crafted, innovative gems. All of the gems are immediately accessible and useful in formulating clean, fast, and elegant programs. The C programming language is used for most of the program listings, although several of the gems have C++ implementations. An IBM or Macintosh disk containing all of the code from all four volumes is included. Includes one 3.5" high-density disk. ~~\$49.95~~ **\$44.95**

How To Write Macintosh Software by Scott Knaster is a great source for understanding Macintosh programming techniques. Drawing from his years of experience working with programmers, Scott explains the mysteries and myths of Macintosh programming with wit and humor. The third edition, fully revised and updated, covers System 7 and 32-bit developments, and explores such topics as how and where things are stored in memory; what things in memory can be moved around and when they may be moved; how to debug your applications with MacsBug; how to examine your program's code to learn precisely what's going on when it runs. 448 pgs., ~~\$29.95~~ **\$26.05**

The Instant Internet Guide by Brent Heslop and David Angell. An Internet jump-start — how to access, use and navigate global networks. The Instant Internet Guide equips readers with the tools needed to travel the electronic world. The book highlights the most important sources of Internet news and information and explains how to access information on remote systems. It outlines how to use essential Internet utilities and programs and includes a primer on UNIX for the Internet. 224 pages ~~\$14.95~~ **\$13.45**

Learn C on the Macintosh by Dave Mark. This self-teaching book/disk package gives you everything you need to begin programming on the Macintosh. Learn to write, edit, compile, and run your first C programs through a series of over 25 projects that build on one another. The book comes with THIN C — a customized version of Symantec's THINK C, the leading programming environment for Macintosh. 464 pages, Book/disk: ~~\$34.95~~ **\$31.45**

Learn C++ on the Macintosh by Dave Mark. After a brief refresher course in C, Learn C++ introduces the basic syntax of C++ and object programming. Then you'll learn how to write, edit, and compile your first C++ programs through a series of programming projects that build on one another as new concepts are introduced. Key C++ concepts such as derived classes, operator overloading, and iostream functions are all covered in Dave's easy-to-follow approach. Includes a special version of Symantec C++ for Macintosh. Book/disk package with 3.5" 800K Macintosh disk. 400 pages, ~~\$36.95~~ **\$33.26**

Macintosh C Programming Primer Volume I, Second Edition, Inside the Toolbox Using THINK C by Dave Mark and Cartwright Reed. This new edition of this Macintosh programming bestseller is updated to include recent changes in Macintosh technology, including System 7, new versions of THINK C and ResEdit, and new Macintosh machines. Readers will learn how to use the resources, Macintosh Toolbox and interface to create stand-alone applications. 672 pages, ~~\$26.95~~ **\$24.25**

Macintosh C Programming Primer Volume II, Mastering the Toolbox Using THINK C by Dave Mark. Volume II picks up where Volume I leaves off, covering more advanced topics such as: Color

QuickDraw, THINK Class Library, TextEdit, and the Memory Manager: 528 pgs. ~~\$26.95~~ **\$24.25**

Macintosh Pascal Programming Primer Volume I, Inside the Toolbox Using THINK Pascal by Dave Mark and Cartwright Reed. This tutorial shows programmers new to the Macintosh how to use the Toolbox, resources, and the Macintosh interface to create stand-alone applications with Symantec's THINK Pascal. 544 pages ~~\$26.95~~ **\$24.25**

Macintosh Programming Techniques by Dan Sydow (Series Editor: Tony Meadow). This tutorial and handbook provides a thorough foundation in the special techniques of Macintosh program-ming for experienced Macintosh programmers as well as those making the transition from DOS, Windows, VAX or UNIX. Emphasizes programming techniques over syntax for better code, regardless of language. Guides the reader through Macintosh memory management, QuickDraw, events and more, using sample program in C++. Disk includes an interactive tutorial, plus reusable C++ code. ~~\$34.95~~ **\$31.95**

NEW! **Multimedia Authoring Building and Developing Documents** by Scott Fisher addresses the concerns that face anyone trying to create multimedia documents. It offers specific advice on when to use different kinds of information architecture, discusses the human-factors concepts that determine how readers use and retain information, and them applies these findings to multimedia documents, covering the high-level issues concerning planners and authors of multimedia documents as well as those involved in evaluating or purchasing multimedia platforms. Includes one 3.5" high-density disk. ~~\$34.95~~ **\$31.45**

NEW! **Programming for the Newton Software Development with NewtonScript** by Julie McKeethan and Neil Rhodes. Foreword by Walter R. Smith. Programming for the Newton: Software Development with NewtonScript is an indispensable tool for Newton programmers. Readers will learn how to develop software for the Newton on the Macintosh from people that developed the course on programming the Newton for Apple Computer. The enclosed 3.5" disk contains a sample Newton application from the books, as well as demonstration version of Newton Toolkit (NTK), Apple Computers complete development environment for the Newtons. A Publication of AP Professional May 1994, Paperback, 393 pp. ~~\$29.95~~ **\$26.95**

Programming in Symantec C++ for the Macintosh by Judy May and John Whittle. This book will introduce you to object-oriented programming, the C++ language, and of course Symantec C++ for the Macintosh. You don't have to be a programmer, or even know anything about programming to benefit from this book. Programming in Symantec C++ for the Macintosh covers everything from the basics to advanced features of Symantec C++. If you are a Think C or Zortech C++ programmer who wants to learn more about object-oriented programming or what's different about Symantec C++, there are whole chapters specifically for you. Includes helpful examples of C++ code that illustrate object-oriented programs. ~~\$29.95~~ **\$26.95**

Programming for System 7 by Gary Little and Tim Swihart, is a hands-on guide to creating applications for System 7. It describes the new features and functions of the operating system in detail. Topics covered include file operations, cooperative multitasking, Balloon Help, Apple events, and the File Manager. Numerous working C code examples show programmers how to take advantage of each of these features and use them in developing their

Want more product info? Call us at 310/575-4343.

applications. 384 pages ~~\$26.95~~ **\$24.25**

ResEdit™ Complete, Second Edition by Peter Alley and Carolyn Strange. With ResEdit, Macintosh programmers can customize every aspect of their interface form creating screen backgrounds and icons to customizing menus and dialog boxes. 608 pages. Book/disk package. ~~\$34.95~~ **\$31.45**

Sad Macs, Bombs, Disasters and What to Do About Them by Ted Landau comes to the rescue with your Macintosh problems. From fractious fonts to the ominous Sad Macintosh icon, this emergency handbook covers the whole range of Macintosh problems: symptoms, causes, and what you can do to solve them. 640 Pages ~~\$24.95~~ **\$22.45**

Software By Design: Creating User Friendly Software by Penny Bauersfeld (Series Editor: Tony Meadow). This excellent reference provides readers with a thorough how-to for designing software that is easy to learn, comfortable to operate and that inspires user confidence. Written from the perspective of Macintosh, but compatible with all platforms. Stresses user input from initial design, through prototyping, testing and revision. Provides tools for analyzing user needs and test responses. Includes exercises for sharpening user-oriented design skills. ~~\$29.95~~ **\$26.95**

NEW! Taligent's Guide to Designing Programs Well-Mannered Object-Oriented Design in C++ is the Taligent approach to object-oriented design. The Taligent Operating Environment is the first commercial software system based entirely on object-oriented technology. Taligent's Guide to Designing Programs is a developer's-eye view of this system. It introduces new concepts of programming and empowers developers to create software more productively. Out of their direct experience in developing the system, the authors focus on global issues of object-oriented design and writing C++ programs, and the specific issues of programming in the Taligent Operating Environment. Taligent's Guide to Designing Programs assumes the reader is an experienced C++ programmer, and proceeds from there to fully explore "the Taligent way" of programming. ~~\$19.50~~ **\$17.55**

Writing Localizable Software for the Macintosh by Daniel R. Carter. 469 pages. ~~\$26.95~~ **\$24.25**

THE APPLE LIBRARY

HyperCard Stack Design Guidelines by Apple Computer, Inc. is an essential book for everyone who creates Apple HyperCard stacks, from beginners to commercial developers. It covers the basic principles of design that, when incorporated, make HyperCard stacks effective and usable. Topics include guidelines, navigation, graphic design and screen illustration, text in stacks, music and sound, a sample stack development scenario, collaborative development, and the Stack Design Checklist. 240 pages. ~~\$21.95~~ **\$19.95**

Inside AppleTalk by Gursharan S. Sidhu, Richard F. Andrews and Alan B. Oppenheimer. Apple Computer, Inc. 650 pages. ~~\$34.95~~ **\$31.45**

Inside Macintosh: AOCCE Application Interfaces by Apple Computer, Inc. shows how your application can take advantage of the system software features provided by PowerTalk system software and the PowerShare collaboration servers. Nearly every Macintosh application program can benefit from the addition of some

of these features. This book shows how you can add electronic mail capabilities to your application, write a messaging application or agent, store information in and retrieve information from PowerShare and other AOCCE catalogs, add catalog-browsing and find-in-catalog capabilities to your application, write templates that extend the Finder's ability to display information in PowerShare and other AOCCE catalogs, add digital signatures to files or to any portion of a document, and establish an authenticated messaging connection. ~~\$40.45~~ **\$36.40**

Inside Macintosh: AOCCE Service Access Modules by Apple Computer, Inc. describes how to write a software module that gives users and PowerTalk-enabled applications access to a new or existing mail and messaging service or catalog service. This book shows how to write a catalog service access module (CSAM), a messaging service access module (MSAM), and AOCCE templates that allow a user to set up a CSAM or MSAM and add addresses to mail and messages. ~~\$26.95~~ **\$24.25**

NEW! Inside Macintosh: CD-ROM by Apple Computer, Inc. Inside Macintosh® is the essential reference for programmers, designers, and engineers for creating applications for the Macintosh family of computers. Inside Macintosh CD-ROM collects more than 25 volumes in electronic form, including: QuickDraw™ GX Library, Macintosh Human Interface Guidelines, PowerPC System Software, Macintosh Toolbox Essentials and More Macintosh Toolbox, QuickTime and QuickTime Components. Now programmers will be able to access over 16,000 pages of the information they need directly from their computers. Hypertext linking and extensive cross referencing across volumes allows programmers to search and explore this library in ways that are unique to the electronic medium. Every Macintosh programmer will regard Inside Macintosh CD-ROM as their most important resource. \$99.95

Inside Macintosh: Devices by Apple Computer, Inc. describes how to write software that interacts with built-in and peripheral hardware devices. With this book, you'll learn how to write and install your own device drivers, desk accessories, and Chooser extensions; communicate with device drivers using the Device Manager; access expansion cards using the Slot Manager; control SCSI devices using SCSI Manager 4.3 or the original SCSI Manager; communicate directly with Apple Desktop Bus devices; interact with the Power Manager in battery-powered Macintosh computers; and communicate with serial devices using the Serial Driver. ~~\$29.95~~ **\$26.95**

Inside Macintosh: Files by Apple Computer, Inc. describes the parts of the operating system that allow you to manage files. It shows how your application can handle the commands typically found in a File menu. It also provides a reference to the File and Alias Managers, the Disk Initialization and Standard File Packages. 510 pgs. ~~\$29.95~~ **\$26.95**

Inside Macintosh: Interapplication Communication by Apple Computer, Inc. shows how applications can work together. How your application can share data, request information or services, allow the user to automate tasks, communicate with remote databases. ~~\$34.95~~ **\$31.45**

Inside Macintosh: Imaging by Apple Computer, Inc. covers QuickDraw and Color QuickDraw. The book includes general discussions of drawing and working with color. It describes the structures that hold images and image information, and the routines that manipulate them. It also covers the Palette, Color, and Printing Managers, and the Color Picker, Color Matching, and Picture

MAIL ORDER STORE

Utilities. ~~\$26.95~~ **\$24.25**

Inside Macintosh: Macintosh Toolbox Essentials by Apple Computer, Inc. covers the heart of the Macintosh. The toolbox enables programmers to create applications consistent with the Macintosh "look and feel". This book describes Toolbox routines and shows how to implement essential user interface elements, such as menus, windows, scroll bars, icons and dialog boxes. 880 pages ~~\$34.95~~ **\$31.45**

Inside Macintosh: More Macintosh Toolbox by Apple Computer, Inc. covers other Macintosh features such as how to support copy and paste, provide Balloon Help, play and record sound and create control panels are covered in this volume. The managers discussed include Help, List, Resource, Scrap and Sound. ~~\$34.95~~ **\$31.45**

Inside Macintosh: Memory by Apple Computer, Inc. describes the parts of the Macintosh operating system that allow you to manage memory. It provides detailed strategies for allocating and releasing memory, avoiding low-memory situations, reference to the Memory Manager, the Virtual Memory Manager, and memory-related utilities. 296 pages. ~~\$24.95~~ **\$22.45**

Inside Macintosh: Networking by Apple Computer, Inc. describes how to write software that uses AppleTalk networking protocols. It describes the components and organization of AppleTalk and how to select an AppleTalk protocol. It provides the complete application interfaces to all AppleTalk protocols, including ATP (AppleTalk Transaction Protocol), DDP (Datagram Delivery Protocol), and ADSP (AppleTalk Data Stream Protocol), among others. ~~\$29.95~~ **\$26.95**

Inside Macintosh: Operating System Utilities by Apple Computer, Inc. describes parts of the Macintosh Operating System that allow you to manage various low-level aspects of the operating system. Everyone who programs the Macintosh should read this book! It will show you in detail how to get information about the operating system, manage operating system queues, handle dates and times, control the settings of the parameter RAM, manipulate the trap dispatch table, and receive and respond to low-level system errors. ~~\$26.95~~ **\$23.45**

Inside Macintosh: Overview by Apple Computer, Inc. is the first book that people who are unfamiliar with Macintosh programming should read. It gives an overview of Macintosh programming fundamentals and a road map to the New Inside Macintosh library. Inside Macintosh: Overview also covers various programming tools and languages, compatibility guidelines and an overview of considerations for worldwide development. 176 pages. ~~\$22.95~~ **\$20.65**

Inside Macintosh: PowerPC Numerics by Apple Computer, Inc. describes the floating-point numerics environment provided with the first release of PowerPC processor-based Macintosh computers. The numerics environment conforms to the IEEE standard 754 for binary floating-point arithmetic. This book provides a description of that standard and shows how RISC Numerics compiles with it. This book also shows programmers how to create floating-point values and how to perform operations on floating-point values in high-level languages such as C and in PowerPC assembly language. ~~\$28.95~~ **\$26.00**

Inside Macintosh: PowerPC System Software by Apple Computer, Inc. describes the new process execution environment and system software services provided with the first version of the system software for Macintosh on PowerPC computers. It

Want more product info? E-mail us at productinfo@xplain.com

MAIL ORDER STORE

contains information you need to know to write applications and other software that can run on the PowerPC. PowerPC System Software shows in detail how to make your software compatible with the new run-time environment provided on PowerPC-based Macintosh computers. It also provides a complete technical reference for the Mixed Mode Manager, the Code Fragment Manager, and the Exception Manager. ~~\$24.95~~ **\$22.45**

Inside Macintosh: Processes by Apple Computer, Inc. describes the parts of the Macintosh operating system that allow you to control the execution of processes and interrupt tasks. It shows in detail how you can use the Process Manager to get information about processes loaded in memory. It is also a reference for the Vertical Retrace, Time, Notification, Deferred Task, and Shutdown Managers. 208 pages, ~~\$22.95~~ **\$20.65**

Inside Macintosh: QuickTime by Apple Computer, Inc. is for anyone who wants to create applications that use QuickTime, the system software that allows the integration of video, animation, and sounds into applications. This book describes all of the QuickTime Toolbox utilities. In addition, it provides the information you need to compress and decompress images and image sequences. ~~\$20.95~~ **\$26.95**

Inside Macintosh: QuickTime Components by Apple Computer, Inc. covers how to use and develop QuickTime components such as image compressors, movie controllers, sequence grabbers, and video digitizers. ~~\$34.95~~ **\$31.45**

Inside Macintosh: Sound by Apple Computer, Inc. describes the parts of the Macintosh system software that allow you to manage sounds. It contains information that you need to know to write applications and other software that can record and play back sounds, compress and expand audio data, convert text to speech, and perform other similar operations. ~~\$26.95~~ **\$24.25**

Inside Macintosh: Text by Apple Computer, Inc. describes how to perform text handling, from simple character display to multi-language processing. The Font, Script, Text Services, and Dictionary Managers are all covered, in addition to QuickDraw Text, TextEdit, and International and Keyboard Resources. ~~\$39.95~~ **\$35.95**

Inside Macintosh: QuickDraw™ GX Library by Apple Computer, Inc. is the powerful new graphics architecture for the Macintosh. Far more than just a revision of QuickDraw, QuickDraw GX is a unified approach to graphics and typography that gives programmers unprecedented flexibility and power in drawing and printing all kinds of shapes, images, and text. This long-awaited extension to Macintosh system software is documented in a library of books that are themselves an extension to the new Inside Macintosh series. The QuickDraw GX Library is clear, concise, and organized by topic. The books contain detailed explanations and abundant programming examples. With extensive cross-references, illustrations, and C-language sample code, the QuickDraw GX Library gives programmers fast and complete reference information for creating powerful graphics and publishing applications with sophisticated printing capabilities. The first two volumes in the QuickDraw GX Library are:

Inside Macintosh: QuickDraw GX Objects by Apple Computer, Inc. introduces QuickDraw GX and its object structure, and shows programmers how to manipulate objects in all types of programs. ~~\$26.95~~ **\$24.25**

Inside Macintosh: QuickDraw GX Graphics by Apple Computer, Inc. shows readers how to create and

manipulate the fundamental geometric shapes of QuickDraw GX to generate a vast range of graphic entities. It also demonstrates how to work with bitmaps and pictures, and specialized QuickDraw GX graphic shapes. ~~\$26.95~~ **\$24.25**

NEW! Inside Macintosh: X-Ref. by Apple Computer, Inc. is an index for Inside Mac. ~~\$12.95~~ **\$11.65**

LANGUAGES



CodeWarrior™ CD by Metrowerks comes into versions — Bronze and Gold. These CDs contain the CodeWarrior development environment including C++, C and Pascal compilers; high-speed linkers; native-mode interactive debuggers; and a powerful new application framework called PowerPlant for rapid Macintosh development in C++. Bronze generates 680x0 code. Gold generates both 680x0 and PowerPC code. All versions are a 3 CD subscription over a 1-year period. Bronze: \$99, Gold: \$399. **Bronze comes with a 6-month MacTech subscription. Gold comes with a 1-year subscription. Both at no additional charge!**



NEW! Geekware by Metrowerks is here! In high school, they called you a computer geek. Now, they work at burger joints and wear polyester uniforms. And you don't. Wear it to your favorite burger joint. \$24.95



FORTRAN by Language Systems is a full-featured ANSI standard FORTRAN 77 compiler that runs in the Macintosh Programmers Workshop (MPW). All major VAX extensions are supported as well as all major features of Cray and Data General FORTRAN. FORTRAN creates System 7 savvy applications quickly and easily. Compiler options specify code generation and optimization for all Macintoshes, including special optimizations for 68040 machines. Error messages are written in plain English and are automatically linked to the source file. The runtime user interface of compiled FORTRAN programs is fully customizable by programmers with any level of Macintosh experience. \$595. w/o MPW: \$495. Corporate 5 pack \$1575

FORTRAN 77 SDK for Power Macintosh by Absoft includes a globally optimizing native compiler and linker, native Fx™ multi-language debugger, and Apple's MPW development environment. The compiler is a full ANSI/ISO FORTRAN 77 implementation and includes all MIL-STD 1753 extensions, Cray/Sun-style POINTER, and several Fortran 90 enhancements. MRWE, Absoft's application framework libraries, is included as is the MIG graphics library for quick creation of plots and graphs. The native Macintosh PPC

toolbox is fully supported. Absoft's Fx debugger can debug intermixed FORTRAN 77, C, C++, PPC assembler. The compiler, linker, and debugger all run as native PPC tools and produce native Macintosh PPC executables. \$699

MacFortran® II V3.3 is a VAX/VMS compatible, full ANSI/ISO FORTRAN 77 compiler including all MIL-STD 1753 extensions. Acknowledged to be the fastest FORTRAN available for Macintosh, MacFortran II is bundled with the latest version of Macintosh Programmer's Workshop (MPW), and includes SourceBug (Apple's source level symbolic debugger) and SoftwareFPU (a math co-processor emulator). Also included is Absoft's Macintosh Runtime Window Environment (MRWE) application framework (with fully documented source code as examples) and MIG graphics library. MacFortran II v3.3 features improved 68040 CPU support and is fully compatible with Power Macintosh under emulation. Documentation includes special sections devoted to use of MacFortran II with the MPW editor and linker, implementation of System 7 features, and porting code to the Macintosh from various mainframes and Unix workstation platforms. \$595



BASIC for the Newton is BASIC for the Newton! From NS BASIC Corporation, it is a fully interactive implementation of the BASIC programming language. It runs entirely on the Newton — no host is required. It includes a full set of functions and data types, hand-written input, windows, buttons and extensions to take advantage of the Newton environment. Applications can create files or access the built-in soups. Applications can also access the serial port for input and output. Work directly on the Newton, or through a connected Mac/PC and keyboard. NS BASIC includes a 150 page pocket sized manual. \$99



SmalltalkAgents™, a superset of the Smalltalk language, is fully integrated with Macintosh, incorporating design features specifically for the RISC and Macintosh System 7 architecture. SmalltalkAgents is a true object oriented workbench that includes an incremental and extensible compiler, an array of design and cross reference tools, pre-emptive interrupt driven threads and events, an extensive class library including classes for general programming, classes for the Macintosh user interface and classes for the Macintosh operating system. Integration of components in enterprise systems is simplified with the network, telecommunication, and inter-application communication libraries. The SmalltalkAgents' extensive class library and add-on components make it especially well suited as a development workbench for custom applications in business, education, science, engineering, and academic research. \$695

SYMANTEC.™

Symantec C++ for Macintosh is an object oriented development environment designed for professional Macintosh programmers. Symantec C++ features powerful object-oriented development tools within a completely integrated environment. The C++ compiler, incremental linker, THINK Class Library, integrated browser, and automatic project management give Symantec C++ fast turnaround times. This product supports multiple editors and translators, so you can use your favorite tools and resource editors as well as scripts

Want more product info? Call us at 310/575-4343.

you've written within the environment. And with ToolServer, you'll be able to customize menus and attach scripts based on Apple events, AppleScript, and MPW Tools. The built-in SourceServer provides a source code control system, allowing teams of programmers to solve tough problems faster. With SourceServer, you'll always know you're working on the latest version. And you'll have old versions at your fingertips when code "breaks" and you need to look back at modifications. Product Contents: Three high density disks, an 832-page user manual, a 568-page THINK Class Library and a 100-page C++ Compiler Guide. \$369

THINK C by Symantec Corporation. THINK C is easy to use and highly visual, making it the No. 1 selling Macintosh programming environment. Enhancements make this product faster and more versatile than ever, improving your productivity with more powerful project management, a full set of tools, and script support for major script-based languages. With the THINK environment, you spend less time on routine programming tasks due to an extremely fast compiler and incremental linker. In addition, the automatic project manager saves you time by tracking changes to your files and recompiling only those that have changes. All the tools you need — a multi-window editor, compiler, linker, debugger, browser, and resource editor — are completely integrated for speed and ease of use. One of the most valuable of these tools is the THINK Class Library, a set of program building blocks that gives you a head start in writing object-oriented applications. And with the new open architecture, you can use your favorite tools, resource editors, and scripts within the environment. THINK C is the logical next step for programmers who have worked in HyperCard or other script-based development environments. The environment supports AppleScript, Apple events, and Frontier, so you can link and automate complex, multi-project operations. Product Contents: Four Macintosh disks, an 832-page user manual, and a 568-page THINK Class Library Guide. \$219

THINK Pascal v. 4.0 by Symantec Corporation. Professionals and students will welcome this version of THINK Pascal. It is fully integrated for rapid turnaround time and lets you take advantage of System 7 capabilities. Features include support for large projects, enhanced THINK Class Library, System 7 compatibility, superior code generation, and smart linking. Product Contents: Four Macintosh disks, a 562-page user manual, and a 498-page object-oriented programming manual. \$169

UTILITIES

NEW! BBEEdit 3.1 from Bare Bones Software is now better than ever. In addition to being Accelerated for Power Macintosh, this powerful, intuitive text editor offers integrated support for THINK C 7.0, Metrowerks CodeWarrior, THINK Reference 2.0 and MPW ToolServer. Version 3.1 adds even more capability, including "soft" wrapping of text on screen and numerous refinements and improvements to the user interface. BBEEdit's many features include: Integrated PopupFuncs(TM) technology for speedy navigation of source code files (C, C++, Pascal, Rez, 68K Assembler, and Fortran), unique 'Find Differences' command (BBEEdit can find differences between projects and folders as well as files), support for Macintosh Drag and Drop for editing and other common tasks, PowerTalk support for reading, sending and composition of PowerTalk mail, scripting via any OSA compatible scripting language including AppleScript and Frontier 3.0, and fast search and replace with optional "grep" matching and multi-file searching. BBEEdit's robust feature set and proven performance and reliability make it the editor of choice for

professionals and hobbyists alike. \$119

C Programmer's Toolbox/MPW Rev. 3.0 by MMCAD. The C Programmer's Toolbox provides a wealth of programming and documentation support tools for developers who are creating new code, porting existing code, or trying to improve and expand existing code. The tools include: CDecl composes and translates C/C++ declaration statements to/from English; CFlow™ determines program function hierarchy, runtime library contents, function/file interdependencies and graphs all or part of a program's functional structure; CHilite™ highlights and prints C/C++ files; CLint™ semantically checks multiple C source files, identifying potential programming bugs; CPrint™ reformats, beautifies and documents C/C++ source files; and more... Works with MPW C/C++, THINK C, requires Apple's MPW. \$295

CLimate by Orchard Software is a command line interface that lets you communicate with your Macintosh using English commands to create, delete, rename, and move files and folders. It can start applications, format disks, restart your computer, etc. CLimate supplements the Finder. It includes a BASIC interpreter that lets you script your Macintosh without AppleScript. The interpreter includes advanced programming constructs: repeat loops, if/then/else conditionals, subroutine calls, etc... CLimate implements wildcard characters, enabling you to work on groups of files. Use CLimate instead of MPW to manage your projects. CLimate is an application occupying 70K disk space. It comes bundled with sample programs and full documentation. \$59.95

CMaster 2.0 by Jersey Scientific installs into THINK C 5 / 6 / 7 and Symantec C++ for Macintosh, and enhances the editor. Use its function popup to select a function and CMaster takes you right to it. Other features include multiple clipboards and markers, a Function Prototyper, and a GoBack Menu which can take you back to previous editing contexts. Almost all features bindable to the keyboard, along over a hundred keyboard-only features like "Add New Automatic Variable." Glossaries, AppleScript and ToolServer support, Macros, and External Tools you create too! \$129.95

Cron Manager by Orchard Software implements the UNIX Cron facility. It can open any Macintosh file on a given date and time. By creating an alias, renaming it to the date and time to open, and moving it into the special Cron Events Folder, Cron Manager will open it. Cron Manager is a control panel that creates the special Cron Events Folder inside your System Folder. It is completely transparent to the user. It works like the Startup Items folder, only smarter. It works with any Macintosh file: if you can double-click to start it, Cron Manager can open it. \$26.95. Cron Manager bundled with CLimate, \$59.95

Dialog Maker by Electric Software Corporation. Migrating from C to C++? Dialog Maker can ease your transition. Dialog Maker is an object-oriented programming library for MPW C++ and Symantec C++ (MPW and Symantec Development Environment versions) which contains a complete set of routines that create a high level interface to dialogs. Dialog Maker provides a small number of simple, yet powerful routines to access and manipulate dialogs. Resources are used to control the most common dialog behavior allowing you to develop your application lightning fast. Minimum requirements System 7.0, MPW 3.2, MPW C++ 3.2, or Symantec C++ 6.0. \$149

dtF is a true relational database system for Apple Macintosh computers. dtF provides a powerful choice for developers who want to create database centered applications with no performance trade-offs. dtF features

MAIL ORDER STORE

SQL, full transaction control, error recovery, single user, client server architecture and multi-platform support including DOS, Windows, OS/2 and UNIX. The C/C++ API is identical and fully portable across all supported platforms. Third-party vendors supporting dtF will be able to offer a variety of advanced features and benefits to their customers royalty free. Tools are included for importing, exporting, creating and managing databases and users. Supported development environments include: Symantec, MPW, MetroWerks and more. Mac/SDK: \$695

InstallerPack™ by StepUp Software is a package of several Installer "atoms" that let developers incorporate graphics, sounds, file compression and custom folder icons into installation scripts. Compression formats supported are Compact Pro & Diamond. Each atom also available separately: \$219

Last Resort Programmer's Edition records every keystroke, command key and mouse event (in local coordinates) to a file on your hard disk. This is especially useful for program testing & debugging, and for technical support and help desks. If something goes wrong (because of a power failure, system crash, forgetting to save or deleting lines) and you lose a word, phrase, or document you can look in the Last Resort keystroke file and recover what you typed. Last Resort is also useful for technical support personnel, when they have to ask "What was the last thing you did before..." \$74.95

LJ Profiler by Lars Jorbebo Datakonsult supports profiling of C++ 68K and Power PC applications compiled with Code Warrior, CFront or SCpp. Based on active profiling, i.e. profiling code called at function enter and exit, the browser application lets you follow call chain timings in hierarchical views or separate windows. Collect, organize, compare and save profiling data from different versions of your application into a project. Scriptable and recordable with full access to most internal data structures. Optional remote profiling and tracking of segment and stack usage. Full source code to what you link into your application. \$295.

LS Object Pascal CD includes the world's first Object Pascal compiler for Power Macintosh. 100% compatible with Apple's MPW Pascal, LS Object Pascal combines the best of Apple's native development tools with innovative new technology developed at Language Systems. Compiler options specify 68K or native PowerPC code generation. Included on the CD are: LS Object Pascal compiler, Universal Pascal Toolbox interfaces, fully loaded MPW 3.3.1, 68K and PowerPC source debuggers, PowerPC assembler, online documentation, Macintosh Tech Notes, and a special version of AppMaker by Bowers Development that generates native Pascal source code. The beta release includes upgrades to v1.0 when it becomes available. \$399

Spellswell 7 1.0.4 is an award-winning, comprehensive, practical spelling checker that works in batch mode or within applications that incorporate the Apple Events Word Services protocol (e.g., Eudora, WordPerfect, Communicate!, and Fair Witness). Spellswell 7 checks for spelling errors as well as common typos like capitalization errors, spaces before punctuation, double double word errors, abbreviation errors, mixed case errors, extra spaces between words, a/an before vowel/consonant, etc... MacTech orders include developer kit with Writesswell Jr., a sample Apple Events Word Services word-processor and its source code. \$74.95

MacAnalyst by Excel Software supports software engineering methods including structured analysis, data modeling, screen prototyping, object-oriented analysis,

Want more product info? E-mail us at productinfo@xplain.com

MAIL ORDER STORE

and data dictionary. This language independent tool is used by system analysts and software designers. Demo \$79, Product \$995

MacAnalyst/Expert by Excel Software supports software engineering methods with the capabilities of MacAnalyst plus state transition diagrams, state transition tables, decision tables and process activation tables. An integrated requirement database provides traceability from requirement statements to analysis or design diagrams, code or test procedures. This tool is well suited to the analysis and design of real-time or requirements driven projects. Demo \$79, Product \$1595

MacDesigner by Excel Software supports software engineering methods including structured design, object-oriented design, data dictionary and code browsing. This tool is well suited to detailed design or maintenance of software development projects. Demo \$79, Product \$995

MacDesigner/Expert by Excel Software supports software engineering methods with the capabilities of MacDesigner plus multi-task design. An integrated requirement database provides traceability from requirement statements to design diagrams, code or test procedures. This tool is well suited to design or maintenance of real-time, multi-tasking software projects. Demo \$79, Product \$1595

MacA&D by Excel Software combines the capabilities of MacAnalyst/Expert and MacDesigner/Expert into a single application. It supports structured analysis and design, object-oriented analysis and design, real-time extensions, task design, data modeling, screen prototyping, code editing and browsing, reengineering, requirement traceability, and a global data dictionary. Demo \$149, Product \$2995



MacWireFrame by Amplified Intelligence. Create your own virtual reality application with MacWireFrame, a virtual reality application frame work. Includes a complete library of object oriented graphics routines, its own easy to understand application frame work (similar to MacApp or TCL but a lot easier to understand), plus an example application program that lets you start solid modeling right away. Comes complete with fully documented source code. All new purchases will be guaranteed a \$49.99 upgrade to the soon to be released, scriptable, MacWireFrame 5.0. Due to the overwhelming response the special price offer has been extended for a little while longer. **Special Offer: \$200.00 \$75!!!!**

McCLint™ Rev. 2.2 by MMCAD. McCLint locates questionable C programming constructs, saving you hours by identifying programming mistakes and latent programming bugs. Some of the checks include variable type usage, conditional and assignment statement usage, arithmetic operations in conditional expressions, misplaced semicolons, pointer type coercion, function argument passing (with and without function prototypes), local and global variable initialization and usage, and existence/shape of return statements. McCLint includes a THINK C like, multiple window editor and source code highlighting system in a fully integrated environment. One or more files can be analyzed in an interactive or batch fashion. Works with THINK C (including OOPS), MPW C,... \$149.95

McCPrint™ Rev 2.2 by MMCAD. McCPrint reformats and beautifies C and C++ source code in a user specified manner. You can transform code to and from your programming style, making source code easier to read and work with. In addition to code formatting, documentation support aids include source code pagination, line number inclusion and control flow

graphing. McCPrint includes a multiple window editor and source code highlighting system in a fully integrated environment. Works with THINK C, MPW C/C++, supports System 7 and 32 bit addressing for use on any system including the Quadras. \$99.95

The Memory Mine™ by Adianta is a stand alone debugging tool for Macintosh and native PowerPC. Programmers can monitor heaps, identify problems such as memory leaks, and stress test applications. Active status of memory in a heap is sampled on the fly: allocation in non-relocatable (Ptr), relocatable (Handle) and free space is shown, as are heap corruption, fragmentation, and more... Allocate, Purge, Compact, and Zap memory let users stress test all or part of a program. Source code is not needed to view heaps. It works on Macintoshes with 68020 or later and System 7.0 or later. \$99

p1 Modula-2 V5.1 is a full implementation of the ISO Standard for Modula-2 which includes exception handling, termination, complex numbers, value constructors, a standard library and more. In addition it supports objects and MacApp, foreign language calls, all current MPW interfaces, optimized 680x0 instructions, three floating point types with four modes of operation, etc. A symbolic window debugger, several utilities and a set of examples (including MacApp tutorial) are included. p1 Modula-2 requires MPW. It is targeted for professional development and prompt technical support by e-mail or FAX is granted. \$395, corporate 5 pack \$1175

NEW! PictureCDEF 1.3 by Paradigm Software is a professional-level CDEF for creating custom graphical buttons (8-64 pixels). PictureCDEF is used in products by Adobe, ProVue, STF Technologies and others. It is multi-monitor and bit-depth sensitive. The button graphic (cicon, ResEdit) can be changed at runtime and even animated with a call-back routine. Create distinct buttons in seven variations: MultiState, PushButton, FlexiButton, ToggleButton, ChkButton, PushPictButton and TogglePictButton. Position the optional button title at left, bottom or right, or allow the system text direction for international support. Manual, sample code and MacApp 3.0 support included. Full source code: \$95.00 Object code: \$45.00.

Qd3d/3dPane/SmartPane source code bundle by Vivistar Consulting. **Qd3d 2.0:** Full featured 3d graphics. Points; lines; polygons; polyhedra; Gouraud shading; z-buffering; culling; depth cueing; parallel, perspective, and stereoscopic projections; performance enhancing "OnlyQD" and "Wireframe" modes; full clipping; pipeline access; animation and model interaction support; and a "triad mouse" to map 2d mouse movement to 3d. **3dPane 2.0:** Integrates Qd3d with the TCL and provides a view orientation controller. **SmartPane:** Offscreen image buffering, flicker free animation, and QuickTime movie recording. For use with Qd3d/3dPane or in 2d settings. All work with C++ compilers or ThinkC 6. \$192

NEW! QC™ by Onyx Technology, is a system extension that stress tests code during runtime for common and not-so-common errors. Tests include heap checks, purges, scrambles, handle/pointer validation, dispose/release checks, write to zero, de-reference zero as well as other tests like free memory invalidation and block bounds checking. QC is extremely user friendly for the non-technical tester yet offers an API for programmers who want precise control over testing. \$99

NEW! QUED/M 2.7 by Nisus Software, is a programmer's text editor which has defined the industry standard for speed and efficiency. With integrated support for Symantec C/C++, Metrowerks CodeWarrior, and MPW, QUED/M offers unrivaled usefulness for the Macintosh developer. In addition to supporting all the major development environments on the Macintosh, QUED/M offers dozens of powerful editing features, including unlimited undo and redo, UNIX style GREP searching, macro language, scripting, text folding, text sorting, file comparison and merging, Toolbox lookup, ten editable/appendable clipboards, line numbering, markers, displaying text as ASCII codes, vertical and horizontal screen splitting, plus much more.

\$149ScriptGen Pro™ by StepUp Software is an Installer script generator which requires no programming or knowledge of Rez. Supports StepUp's InstallerPack, StuffIt compression, custom packages, splash screens, network installs, Rez code output, importing resources, and AppleEvent link w/MPW: \$169

SoftPolish by Language Systems is a development tool that helps software developers avoid embarrassing spelling errors, detect incorrect or incompatible resources and improve the appearance of their Macintosh software. SoftPolish examines application resources and reports potential problems to a scrolling log. Independent of any programming language or environment, SoftPolish improves the quality of any Macintosh program. \$169

NEW! Spyer by InCider is a simple operated tool that records all actions (including mouse movement) you perform on a Macintosh computer and then replays them at your preferred speed. The recorded data can be saved in files for future use. Spyer works as a background process with any Macintosh application and is triggered by user defined Hot Keys. Spyer enables the "Continuous Redo" utility and is especially useful for software testing and demonstration. \$39

StoneTable: A library replacing all functions found in list manager plus: variable size columns/rows; different font, size, style, foreground, background per cell; sort, resize, move, copy, hide columns/rows; edit cells/titles in place; titles for columns/rows; multiple lines per cell; grid line pattern/color; greater than 32k data per table; up to 32k text per cell; support for balloon help and binary cell data. Versions for Think C, Think Pascal, MPW C, MPW Pascal, CodeWarrior C. (all prices per developer) \$150 first compiler, additional compilers \$50

Stone Table Extra: Additional functions for StoneTable. Drag selected cells within table or to other tables; optionally add rows as part of drag; popup menus or check boxes in cells; variable width grid lines; move/drag/resize table in window; clipboard operations on multiple cells. Requires StoneTable. (all prices per developer) \$50 first compiler, additional compilers \$25

NEW! StoneTable and StoneTableExtra for PowerPC: Same functionality as 68K libraries. Versions for MPW C and CodeWarrior C. Must have 68K libraries. (all prices per developer) StoneTable \$100, StoneTableExtra \$25

ViperBase by Viper Development is a fast database designed for developers that want speed but don't want to spend months or years developing a commercial quality database. ViperBase: Unlimited Records, Variable Length: \$59. ViperBase II: ViperBase + Multiple Indices. \$119



Want more product info? Call us at 310/575-4343.

LIST OF ADVERTISERS

Absoft	43
Adianta Inc.	24
Aladdin Knowledge Systems Ltd.	5
APDA, Apple Computer, Inc.	19, 45
Apple Developer University	57
BareBones Software	49
Celestin Company	56
Creative Solutions	28
DataPak Software, Inc.	54
Douglas Electronics, Inc.	53
Dell	40
Excel Software	65
Foundation Solutions	31
Full Moon Software	47
Graphic Magic	62
Graphical Business Interfaces Inc.	26 & 27
Green Dragon Creations, Inc.	52
Jasik Designs	17
Language Systems	22
Logic Programming Associates, Ltd.	32
MacTech CD-ROM, Vol. 1-9	72
MacXperts	71
Mainstay	1
Manpower Technical Services	71
Mathemaesthetics, Inc.	13
Metrowerks	6
Microsoft Corporation	37-39
MindVision Software	21
Neologic Systems	25
Nisus	23
Onyx Technology	58
Options Computer Consulting	50
PACE Anti-Piracy	34
Prograph International	IFC
Quasar Knowledge Systems	IBC
Rainbow Technologies	15
Ray Sauers Associates	68
Richey Software Training	59
Scientific Placement	71
Sierra Software Innovations	BC
SNA, Inc.	61
StepUp Software	62
Summit Software Company	55
The Mac Group	60
TSE International	29
Userland Software, Inc.	11
Vermont Database Corporation	51
Water's Edge Software	64

LIST OF PRODUCTS

Apple's Installer 4.0 • StepUp Software	62
AppleScript™ • APDA, Apple Computer, Inc.	19
Apprentice • Celestin Company	56
BasicScript • Summit Software Company	55
BEdit 3.1 • BareBones Software	49
4D TOOLKIT 2.0 • Options Computer Consulting	50
C++ For Power Macintosh • Absoft	43
Cataloger™ • Graphical Business Interfaces Inc.	26 & 27
CodeWarrior™ • Metrowerks	6
Custom Software Development • GreenDragon Creations, Inc.	52
CXBase Pro • TSE International	29
The Debugger V2 • Jasik Designs	17
Debugging Services • The Mac Group	60
The Dell® Developer System • Dell	40
Developer Vise 3.0 • MindVision Software	21
DragInstall 1 • Ray Sauers Associates	68
EHelp 4.0 • Foundation Solutions	31
E.T.O. • APDA, Apple Computer, Inc.	19
Frontier 3.0 • Userland Software, Inc.	11
HyperCard® 2.2 • APDA, Apple Computer, Inc.	19
IcePick 3.0™ • Sierra Software Innovations	BC
Inside Out II® • Sierra Software Innovations	BC
LS Object Pascal/PPC • Language Systems	22
MacAnalyst • Excel Software	65
MacApp Programmers • MacXperts	71
MacDesigner • Excel Software	65
MacDisk Duplicator • Douglas Electronics, Inc.	53
MacEncrypt™ • PACE Anti-Piracy	34
MacForth Plus 4.2 • Creative Solutions	28
MacHASP • Aladdin Knowledge Systems Ltd.	5
MacNosy • Jasik Designs	17
Mac OS SDK • APDA, Apple Computer, Inc.	45
MacProlog32 • Logic Programming Associates, Ltd.	32
MacRegistry™ • Scientific Placement	71
MPW® Pro • APDA, Apple Computer, Inc.	19
neoAccess™ • Neologic Systems	25
OP2CPlus • Graphic Magic	62
p2cII™ • Sierra Software Innovations	BC
PAIGE™ • DataPak Software, Inc.	54
PatchWorks™ • SNA, Inc.	61
Power MacForth • Creative Solutions	28
Power PC Training • Apple Developer University	57
Prograph CPX • Prograph International	IFC
Programmer Training • Richey Software Training	59
QC: The Macintosh Testing Solution • Onyx Technology	58
QUED/M 2.7 • Nisus	23
RC/21 • Vermont Database Corporation	51
Recruitment • MacXperts	71
Recruitment • Manpower Technical Services	71
Recruitment • Scientific Placement	71
Resorcerer® 1.2 • Mathemaesthetics, Inc.	13
SALESBASE 2.1™ • Sierra Software Innovations	BC
ScriptGen Pro 2.0 • StepUp Software	62
ScriptWizard • Full Moon Software	47
Sentinel® • Rainbow Technologies	15
Sierra Consulting Group • Sierra Software Innovations	BC
SmalltalkAgents® • Quasar Knowledge Systems	IBC
Template Constructor™ • Graphical Business Interfaces Inc.	26 & 27
The Memory Mine™ • Adianta Inc.	24
Tool Plus™ 2.5 • Water's Edge Software	64
VIP-C 1.5 • Mainstay	1
Windows™ 95 • Microsoft Corporation	37-39

By Scott T Boyd, Editor



TIP OF THE MONTH

KEEPING THE PORTS STRAIGHT

Saving and restoring the GrafPort is something that must be done a lot. Suppose you needed to convert a Point to Local coordinates. You first have to set the GrafPort to the window who's coordinates you want. It might look like this:

```
void foo(WindowPtr myWindow, Point *thePoint) {
    GrafPtr savedPort; // declare a temporary variable
    GetPort(&savedPort); // remember the old port
    SetPort(myWindow); // point to the right window
    GlobalToLocal(thePoint); // do the work
    SetPort(savedPort); // restore the old port
}
```

Here's a nifty C++ class to simplify things. Put it all in one header file, PortSaver.h:

```
class PortSaver {
public:
    PortSaver(GrafPtr newPort = nil);
    virtual ~PortSaver(void);
private:
    GrafPtr savedPort;
};
inline PortSaver::PortSaver(GrafPtr newPort) {
    GetPort(&savedPort); // remember the old port
    if (newPort != nil)
        SetPort(newPort); // set the new port
}
inline PortSaver::~~PortSaver(void) {
    if (savedPort != nil) // if there is an old port
        SetPort(savedPort); // restore it
}
// A macro that makes the PortSaver easier to use
#define SETPORT(aPort) PortSaver setCurrentPortTo(aPort)
```

Here is the same routine using class PortSaver

```
void foo(WindowPtr myWindow, Point *thePoint) {
    SETPORT(myWindow); // point to the right window
    GlobalToLocal(thePoint); // do the work
}
```

Advantages:

- 1) Saves typing. Actually, class PortSaver does more sanity checking than the simple example above.
- 2) The port is always restored when the function exits, no matter how many return points there might be, and even if an exception gets thrown.

Disadvantages

Even though I've written PortSaver with inline functions, there is no guarantee that any given compiler will actually inline them. As a result, you might get additional overhead from the function calls, and a call to LoadSeg (depending on where the compiler actually put the function code.)

You can adapt PortSaver to save other things as well. Whenever you want to temporarily change one of the global Quickdraw properties (text font, window origin, etc.), you can implement a PortSaver-like class.

— James Jennings, jennings@balcyon.com

You may already have won! But you'll never know if you don't send us your tips! We pay \$25 for every tip we use, except for Tip of the Month, which scores twice as much. You can take your award in orders, subscriptions or US dollars. Make sure your code compiles, and send tips (and where to mail the moolah) to editorial@xplain.com. See page two for our other addresses.

OK, EVERYBODY, GET IN LINE!

The review of Object Master 2.5 in the December issue was a welcome insight into a powerful programming tool. On page 65, Andy Dent observes that Object Master allows both color- and style-coding of source files, and points out that this feature is a "contentious issue." Andy is probably referring to the fact that "styling" code often results in defeating the uniformity of text width in mono spaced fonts that most programmers rely on to align code vertically.

I've used Object Master, and found that you can have code styled and mono spaced by using the "condensed" style in conjunction with "bold" or "outline" style. Italics and plain text characters are then the same width as bold or outlined characters. As a result, code text is dramatically clearer when read with a styled editor like Object Master, but loses none of its structure when read by an editor that doesn't support styled text.

— Nick, nick+@pitt.edu

SmalltalkAgents®

A Dynamic Object-Oriented Development Environment

Agents Object System (AO/S): Delivering Component-based Technology

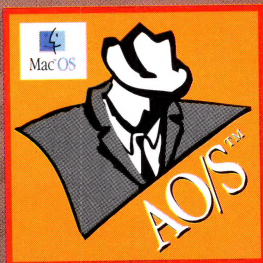
SmalltalkAgents is a set of development and authoring components built on the Agents Object System™ (AO/S™). The AO/S Component Toolbox™ is a portable layer of abstraction between AO/S Components and host system services. The AO/S delivers a user extensible home/container that transparently wrappers a variety of component technologies including OpenDoc, OLE, and OSA Scripting.

AO/S: Core Technologies

The AO/S Core Technologies include rich intelligent component and agent support, vendor independent database services, concurrent application support (much like the Apple Finder), separate user interface threads enabling both tethered development and robust "memory protected" application deployment, and a shareable object system, all of which makes it ideal for high-performance client and server applications.

Scalability

Unlike other "application builders", the AO/S Product Family (including SmalltalkAgents Professional and VisualAgents) allows you to go from user-level scripting to full professional-level system development and back. SmalltalkAgents is scalable, from the creation of small and simple applications to sophisticated and complex multi-user systems.



Agents Object System
MACINTOSH

Call now to receive our
customer's own
descriptions of why
they're choosing AO/S to
deliver their products!

Based on industry- proven Smalltalk language

Smalltalk is the pure Object-Oriented (OO) language which defined the OOP concept, and is now the corporate language of choice for new business applications and sophisticated client/server systems.

What this means to you!

SmalltalkAgents Professional, built on the unique AO/S architecture, is a modern, innovative new development system that blends the best of component-based architecture with the best in object-oriented development technologies.

What the press and customers say:

"..... the most unique and innovative
Smalltalk development environment..."

Object Magazine, October 1994

"I must say, this is NOT the Smalltalk of several years ago, and I AM TRULY IMPRESSED WITH WHAT YOU HAVE ACCOMPLISHED!"

E.S. Taylor, Independent Consultant, New York, NY

"SmalltalkAgents has been my favorite Smalltalk environment and I believe it represents the best chance Smalltalk has of becoming an accepted grass roots language for developers."

David Scott, General Atomics, San Diego, CA

Don't be passed by, catch the Component
Technology wave with SmalltalkAgents! To gain
the Power Macintosh advantage now, call:

1-800-296-1339
or 301-530-4853 (info@qks.com)



Buy now to receive free upgrade
to SmalltalkAgents Pro v2.0!



Sierra Software Innovations

Six Years of Quality Products and Services

SALESBASE 2.1™

Fully Integrated Sales Automation Solution for Large Corporations

Inside Out II®

Industrial Strength Multi-User Relational Database Engine

p2cII™

Advanced Object Pascal to C++ Source Code Translator

IcePick 3.0™

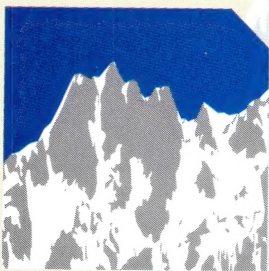
Original Award Winning MacApp 2.x and 3.x View Editor

Sierra Consulting Group

Top-Notch DataBase and SalesAutomation Consulting

Please call 800.621.0631 for a complete information package
on Sierra's line of prestigious products and custom services.

Better Software through Innovation



**SIERRA
SOFTWARE
INNOVATIONS**

923 Tahoe Blvd., Incline Village, Nevada 89451

Phone: 702.832.0300 Fax: 702.832.7753 Internet: D2086@applelink.apple.com

SALESBASE, Inside Out II, p2cII, IcePick are trademarks of Sierra Software Innovations.